

## Chapter 2 : Introduction to Programming Languages , Programming Fundamentals, Java Operators

### INTRODUCTION TO PROGRAMMING

#### PROGRAMMING FUNDAMENTALS

Java programming language was originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform (Java 1.0 (J2SE)). As of December 2008, the latest release of the Java Standard Edition is 6 (J2SE). With the advancement of Java and its widespread popularity, multiple configurations were built to suite various types of platforms. Ex: J2EE for Enterprise Applications, J2ME for Mobile Applications. Sun Microsystems has renamed the new J2 versions as Java SE, Java EE and Java ME respectively. Java is guaranteed to be **Write Once, Run**

**Anywhere. Java is:**

- **Object Oriented:** In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
- **Platform independent:** Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by virtual Machine (JVM) on whichever platform it is being run.
- **Simple:** Java is designed to be easy to learn. If you understand the basic concept of OOP Java would be easy to master.
- **Secure:** With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
- **Architectural-neutral :**java compiler generates an architecture-neutral object file format which makes the compiled code to be executable on many processors, with the presence of Java runtime system.
- **Portable:** Being architectural-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary which is a POSIX subset.
- **Robust:** Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.
- **Multithreaded:** With Java's multithreaded feature it is possible to write programs that can do many tasks simultaneously. This design feature allows developers to construct smoothly running interactive applications.
- **Interpreted:** Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an weight process.
- **High Performance:** With the use of Just-In-Time compilers, Java enables high performance.
- **Distributed:** Java is designed for the distributed environment of the internet.
- **Dynamic:** Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information verify and resolve accesses to objects on run-time.

#### Programming Fundamentals:-

Token:- The smallest individual unit in a program is known as Token. Java has the following types of tokens:-Keyword ,Identifier, Literals, Punctuators and operators.

**Keywords:-**The following list shows the reserved words in Java. These reserved words may not be used as constant or variable or any other identifier names.

abstract	assert	boolean	break
byte	case	catch	char
class	const	continue	default
do	double	else	enum
extends	final	finally	float
for	goto	if	implements
import	instanceof	int	interface
long	native	new	package
private	protected	public	return
short	static	strictfp	super
switch	synchronized	this	throw
throws	transient	try	void
volatile	while		

#### Literals:-

A literal is a source code representation of a fixed value. They are represented directly in the code without any computation. Literals can be assigned to any primitive type variable. For example:

```
byte a = 68;  
char a = 'A'
```

byte, int, long, and short can be expressed in decimal(base 10), hexadecimal(base 16) or octal(base 8) number systems as well. Prefix 0 is used to indicate octal and prefix 0x indicates hexad these number systems for literals. For example:

```
int decimal = 100;  
int octal = 0144;  
int hexa = 0x64;
```

String literals in Java are specified like they are in most other languages by enclosing a sequence of characters between a pair of double quotes. Examples of string literals are:

```
"Hello World"  
"two\nlines"  
"\"This is in quotes\""
```

String and char types of literals can contain any Unicode characters. For example:

```
char a = '\u0001';  
String a = "\u0001";
```

#### Identifiers:-

All Java components require names. Names used for classes, variables and methods are called identifiers. In Java, there are several points to remember about identifiers. They are as follows:

- All identifiers should begin with a letter (A to Z or a to z), currency character ( \$ or % ) or underscore ( \_ ).
- After the first character identifiers can have any combination of characters.
- A keyword cannot be used as an identifier.
- Most importantly identifiers are case sensitive.
- Examples salary, \_value, \_\_1\_value
- Examples of illegal identifiers: 123abc, -salary

#### Java Operators

Java provides a rich set of operators to manipulate variables. We can divide all the Java operators into the following groups:

- Arithmetic Operators
- Relational Operators
- Bitwise Operators

- Logical Operators
- Assignment Operators
- Misc Operators

### The Arithmetic Operators:

Arithmetic operators are used in mathematical expressions in the same way that they are used in algebra. The following table lists the arithmetic operators: Assume integer variable A holds 10 and variable B holds 20, then:

Operator	Description	Example
+	Addition - Adds values on either side of the operator	A + B will give 30
-	Subtraction - Subtracts right hand operand from left hand operand	A - B will give -10
*	Multiplication - Multiplies values on either side of the operator	A * B will give 200
/	Division - Divides left hand operand by right hand operand	B / A will give 2
%	Modulus - Divides left hand operand by right hand operand and returns remainder	B % A will give 0
++	Increment - Increases the value of operand by 1	B++ gives 21
--	Decrement - Decreases the value of operand by 1	B-- gives 19

### The Relational Operators:

There are following relational operators supported by Java language Assume variable A holds 10 and variable B holds 20, then:

Operator	Description	Example
==	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

### The Logical Operators:

The following table lists the logical operators: Assume Boolean variables A holds true and variable B holds false, then:

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B) is false.
	Called Logical OR Operator. If any of the two operands are non-zero, then the condition becomes true.	(A    B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is true.

### The Assignment Operators:

There are following assignment operators supported by Java language:

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assign value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A
<<=	Left shift AND assignment operator	C <<= 2 is same as C = C << 2
>>=	Right shift AND assignment operator	C >>= 2 is same as C = C >> 2
&=	Bitwise AND assignment operator	C &= 2 is same as C = C & 2
^=	bitwise exclusive OR and assignment operator	C ^= 2 is same as C = C ^ 2
=	bitwise inclusive OR and assignment operator	C  = 2 is same as C = C   2

### Misc Operators

There are few other operators supported by Java Language.

**Conditional Operator ( ? : ) :-** Conditional operator is also known as the ternary operator. This operator consists of three operands and is used to evaluate Boolean expressions. The goal of decide which value should be assigned to the variable. The operator is written as:

variable x = (expression) ? value if true : value if false

Following is the example:

```
int a, b;
a = 10;
b = (a == 1) ? 20 : 30;
System.out.println("Value of b is : " + b);
b = (a == 10) ? 20 : 30;
System.out.println("Value of b is : " + b);
```

This would produce the following result:

```
Value of b is : 30
Value of b is : 20
```

## Unit 2 Part 2 Flow of Control in C, C++, Java, and Code with programming

### Software Concepts & Productivity Tools

An ordered set of instructions given to the computer is known as a program and a set of such programs that governs the operation of a computer system and/or its related devices is known as Software.

#### Types of Software:

##### System Software:

Software can be divided into different types depending upon their uses and application- System Software & Application Software.

Software required to run and maintain basic components of computer system come under the category of system software whereas software required to solve some specific task of daily use is generally called **application software**.

An operating system is an example of system software while documentation tool, a presentation tool, a spreadsheet tool are all examples of application software. Even your favorite computer game is an example of application software. Some common examples of System Software as follows :

##### 1. BIOS :-

The basic input/output system (BIOS) is also commonly known as the System BIOS. The BIOS is boot firmware, a small program that controls various electronic devices attached to the main computer system. The BIOS sets the machine hardware into a known state to help the operating system to configure the hardware components. This process is known as booting, or booting up. BIOS programs are stored on a chip

## **2. Operating System :-**

Operating system is a set of system programs that controls and coordinates the operations of a computer system.

Operating systems perform all basic tasks, such as identifying basic input/output devices, accepting input from the input devices, sending results to the output devices, keeping track of files and directories on the disk, and controlling other peripheral devices such as disk drives and printers

### **Need for an Operating System**

Operating system provides a software platform, on top of which, other programs, called application programs are run.

### **Major Functions of an Operating System**

The functions of an operating system can be broadly outlined as follows:

- Communicate with hardware and the attached devices [Device Manager]
- Manage different types of memories [Memory Manager]
- Provide a user interface [Interface Manager]
- Provide a structure for accessing an application [Program Manager]
- Enable users to manipulate programs and data [Task Manager]
- Manage the files, folders and directory systems on a computer [File Manager]

Following types of operating system are generally available and used depending upon the primary purpose and application and the type of hardware attached to the computer:

#### **Single User :-**

Allows one user to operate the computer and run different programs on the computer. MS DOS is a common example of single user operating system.

#### **Multi-user :-**

Allows two or more users to run programs at the same time on a single computer system. Unix, Linux, Windows are

common examples of multi user operating system.

**Real Time:-**

Responds to input instantly. Real-time operating systems are commonly found and used in robotics, complex multimedia and animation, communications and has various military and government uses. LYNX and Windows CE are examples of real time operating systems.

**3.Device Driver :-**

A device driver is a system software that acts like an interface between the Device and the user or the Operating System. All computer accessories like Printer, Scanner, Web Camera, etc come with their own driver software.

**4. Language Processor :-**

A computer system understands only machine language or binary language, also known as Low Level Language(LL). This language is extremely difficult to learn for a general programmer and thus there is a need for some special language that is easy to learn and understand for the programmer in order to interact with the computer system. These special languages or set of commands are collectively known as programming languages or High Level languages (HLL).

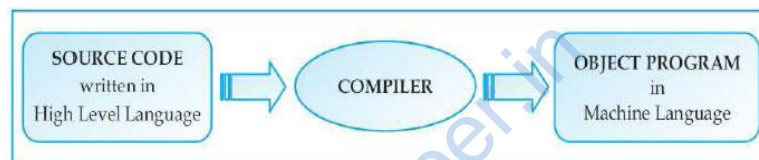
Some examples of High Level Programming

Languages are Basic, C, C++, JAVA, etc. These high level programming languages can easily be translated into machine language using Language Processors. These are:

**Assembler :-** Assembler is a language processor, which translates a program written in assembly language into machine language.

**Compiler :-** A compiler is a language processor which converts (or translates) the entire program written in high level language into machine language in one go.

**Interpreter :-**This language processor converts a high level language program into machine language line by line as well as executes it.



*Conversion of Source Code into Object Code*

www.dreamtopper.in

### Application Software :-

Application software is a set of programs to carry out a specific task like word processor, spreadsheet, presentation tools, library management software, railway reservation, antivirus software, etc. Application Software can be divided into different categories depending upon their uses as follows:

- Utility Software
- General Purpose Application Software
- Specific Purpose Application Software
- Developer Tools

### Utility Software

We also require some additional software to keep our computer system efficient and trouble free. Generally these software come bundled with the Operating System Software but we can also use utility software provided by other vendors. Few examples of utility software are as follows:

- **Compression utility software :-** Using this software, you can reduce (compress) the storage size of any computer program/file while not in use.
- **Backup utility software :-** Though computer is in general a dependable device but it is always advisable to take regular back up of important data and programs stored in the computer. In case of any damage to the system, the back-up files can be restored and the important data can be recovered from the back-up files. This utility software facilitates you to take regular back-up of important files and folders stored in a drive into another storage device
- **Disk De-fragmentation Utility software :-** When computer system finds a file too large to store in a single location, it splits the file and stores it in pieces (called fragments), which are logically linked. This simply means that different parts of the file are scattered across the hard drive in noncontiguous locations. Disk de-fragmentation utility software speeds up the system by rearranging such fragmented files stored on a disk in contiguous locations in order to optimize the system performance.
- **Antivirus detection and protection software :-** This utility software provides the user with a virus free work environment by restricting the entry of any unwanted program into the system.
- **Text Editor :-** This utility software helps one to create, store or edit a basic text file examples of text editors are Notepad, Gedit and K Write

### General Purpose Application Software

Some of the application software are designed for general day to day applications and uses. Some of these popular general

purpose application software are discussed below: Word

**Processor :-** Word Processor is general purpose application software that facilitates the creation of text documents with extensive formatting.

**Spreadsheet Tools :-** Spreadsheet Tool is general purpose application software that facilitates creation of tabular forms where some text and numerical values can be stored .

**Database Management System :-** Database Management System is general-purpose application software that facilitates creation of computer programs that control the creation, maintenance, and the use of database for an organization and its end users.

**Specific Purpose Application Software Some :-** application software are made for performing specific tasks generally used by the institutions, corporate, business houses, etc. e.g.

**Inventory Management System & Purchasing System :-** Inventory Management System is generally used in departmental stores or in an institution to keep the record of the stock of all the physical resources.

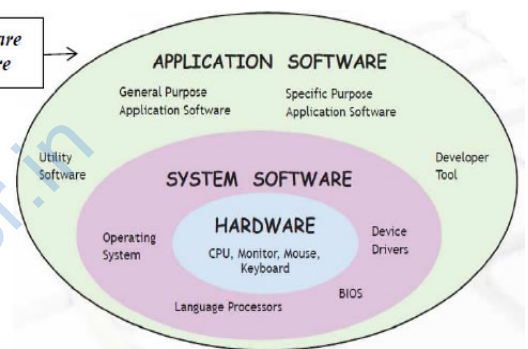
**Developer tools :-** When a programmer starts the process of writing a program to develop software for any type of application, he/she requires a series of software developing tools like code editor, debugger and compiler. Integrated Development Environment An

### Integrated Development Environment

(IDE) is an application program that consists of all required software developing tools required for developing software as part of a single interface. It typically consists of the following tools:

- Source Code Editor
- Graphical User Interface (GUI) builder
- Compiler / Interpreter
- Debugger
- Build Automation tool

*Relationship between Hardware and different types of Software*



www.dreamtopper.in

## UNIT – 2

### Introduction to Programming

#### PROGRAMMING FUNDAMENTALS

Java programming language was originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform (Java 1.0 [J2SE]). As of December 2008, the latest release of the Java Standard Edition is 6 (J2SE). With the advancement of Java and its widespread popularity, multiple configurations were built to suite various types of platforms. Ex: J2EE for Enterprise Applications, J2ME for Mobile Applications. Sun Microsystems has renamed the new J2 versions as Java SE, Java EE and Java ME respectively. Java is guaranteed to be **Write Once, Run Anywhere**.

#### **Java is:**

- **Object Oriented:** In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
- **Platform independent:** Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by virtual Machine (JVM) on whichever platform it is being run.
- **Simple:** Java is designed to be easy to learn. If you understand the basic concept of OOP Java would be easy to master.
- **Secure:** With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
- **Architectural-neutral :**Java compiler generates an architecture-neutral object file format which makes the compiled code to be executable on many processors, with the presence of Java runtime system.
- **Portable:** Being architectural-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary which is a POSIX subset.
- **Robust:** Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.
- **Multithreaded:** With Java's multithreaded feature it is possible to write programs that can do many tasks simultaneously. This design feature allows developers to construct smoothly running interactive applications.
- **Interpreted:** Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light weight process.
- **High Performance:** With the use of Just-In-Time compilers, Java enables high performance.
- **Distributed:** Java is designed for the distributed environment of the internet.
- **Dynamic:** Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.



## Programming Fundamentals:-

**Token:-** The smallest individual unit in a program is known as Token. Java has the following types of tokens:-Keyword ,Identifier, Literals, Punctuators and operators.

**Keywords:-**The following list shows the reserved words in Java. These reserved words may not be used as constant or variable or any other identifier names.

abstract	assert	boolean	break
byte	case	catch	char
class	const	continue	default
do	double	else	enum
extends	final	finally	float
for	goto	if	implements
import	instanceof	int	interface
long	native	new	package
private	protected	public	return
short	static	strictfp	super
switch	synchronized	this	throw
throws	transient	try	void
volatile	while		

**Literals:-** A literal is a source code representation of a fixed value. They are represented directly in the code without any computation.

Literals can be assigned to any primitive type variable. For example:

```
byte a = 68;  
char a = 'A'
```

byte, int, long, and short can be expressed in decimal(base 10), hexadecimal(base 16) or octal(base 8) number systems as well.

Prefix 0 is used to indicate octal and prefix 0x indicates hexadecimal when using these number systems for literals. For example:

```
int decimal = 100;  
int octal = 0144;  
int hexa = 0x64;
```

String literals in Java are specified like they are in most other languages by enclosing a sequence of characters between a pair of double quotes. Examples of string literals are:

```
"Hello World"  
"two\nlines"  
"\\"This is in quotes\""
```

String and char types of literals can contain any Unicode characters. For example:

```
char a = '\u0001';  
String a = "\u0001";
```

**Identifiers:-**All Java components require names. Names used for classes, variables and methods are called identifiers.

In Java, there are several points to remember about identifiers. They are as follows:

- All identifiers should begin with a letter (A to Z or a to z), currency character (\$) or an underscore (\_).
- After the first character identifiers can have any combination of characters.
- A key word cannot be used as an identifier.
- Most importantly identifiers are case sensitive.
- Examples of legal identifiers: age, \$salary, \_value, \_\_1\_value
- Examples of illegal identifiers: 123abc, -salary

### Java Operators

Java provides a rich set of operators to manipulate variables. We can divide all the Java operators into the following groups:

- Arithmetic Operators
- Relational Operators
- Bitwise Operators
- Logical Operators
- Assignment Operators
- Misc Operators

#### The Arithmetic Operators:

Arithmetic operators are used in mathematical expressions in the same way that they are used in algebra. The following table lists the arithmetic operators:

Assume integer variable A holds 10 and variable B holds 20, then:

Operator	Description	Example
+	Addition - Adds values on either side of the operator	A + B will give 30
-	Subtraction - Subtracts right hand operand from left hand operand	A - B will give -10
*	Multiplication - Multiplies values on either side of the operator	A * B will give 200
/	Division - Divides left hand operand by right hand operand	B / A will give 2
%	Modulus - Divides left hand operand by right hand operand and returns remainder	B % A will give 0
++	Increment - Increases the value of operand by 1	B++ gives 21
--	Decrement - Decreases the value of operand by 1	B-- gives 19

#### The Relational Operators:

There are following relational operators supported by Java language

Assume variable A holds 10 and variable B holds 20, then:

Operator	Description	Example
==	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

### The Logical Operators:

The following table lists the logical operators:

Assume Boolean variables A holds true and variable B holds false, then:

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B) is false.
	Called Logical OR Operator. If any of the two operands are non-zero, then the condition becomes true.	(A    B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is true.

### The Assignment Operators:

There are following assignment operators supported by Java language:

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assign value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A

/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A
<<=	Left shift AND assignment operator	C <<= 2 is same as C = C << 2
>>=	Right shift AND assignment operator	C >>= 2 is same as C = C >> 2
&=	Bitwise AND assignment operator	C &= 2 is same as C = C & 2
^=	bitwise exclusive OR and assignment operator	C ^= 2 is same as C = C ^ 2
=	bitwise inclusive OR and assignment operator	C  = 2 is same as C = C   2

### Misc Operators

There are few other operators supported by Java Language.

**Conditional Operator ( ? : ) :-** Conditional operator is also known as the ternary operator. This operator consists of three operands and is used to evaluate Boolean expressions. The goal of the operator is to decide which value should be assigned to the variable. The operator is written as:

variable x = (expression) ? value if true : value if false

Following is the example:

```
int a , b;
a = 10;
b = (a == 1) ? 20: 30;
System.out.println( "Value of b is : " + b );
b = (a == 10) ? 20: 30;
System.out.println( "Value of b is : " + b );
```

This would produce the following result:

```
Value of b is : 30
Value of b is : 20
```

### Data Types:-

There are two data types available in Java:

- Primitive Data Types
- Reference/Object Data Types

#### **Primitive Data Types:**

There are eight primitive data types supported by Java. Primitive data types are predefined by the language and named by a keyword. Let us now look into detail about the eight primitive data types.

**byte**

- Byte data type is an 8-bit signed two's complement integer.
- Minimum value is -128 ( $-2^7$ )
- Maximum value is 127 (inclusive) ( $2^7 - 1$ )
- Default value is 0
- Byte data type is used to save space in large arrays, mainly in place of integers, since a byte is four times smaller than an int.
- Example: byte a = 100 , byte b = -50

**short**

- Short data type is a 16-bit signed two's complement integer.
- Minimum value is -32,768 ( $-2^{15}$ )
- Maximum value is 32,767 (inclusive) ( $2^{15} - 1$ )
- Short data type can also be used to save memory as byte data type. A short is 2 times smaller than an int
- Default value is 0.
- Example: short s = 10000, short r = -20000

**int**

- int data type is a 32-bit signed two's complement integer.
- Minimum value is - 2,147,483,648. ( $-2^{31}$ )
- Maximum value is 2,147,483,647 (inclusive). ( $2^{31} - 1$ )
- int is generally used as the default data type for integral values unless there is a concern about memory.
- The default value is 0.
- Example: int a = 100000, int b = -200000

**long**

- Long data type is a 64-bit signed two's complement integer.
- Minimum value is -9,223,372,036,854,775,808. ( $-2^{63}$ )
- Maximum value is 9,223,372,036,854,775,807 (inclusive). ( $2^{63} - 1$ )
- This type is used when a wider range than int is needed.
- Default value is 0L.
- Example: long a = 100000L, int b = -200000L

**float**

- Float data type is a single-precision 32-bit IEEE 754 floating point.
- Float is mainly used to save memory in large arrays of floating point numbers.
- Default value is 0.0f.
- Float data type is never used for precise values such as currency.
- Example: float f1 = 234.5f

**double**

- double data type is a double-precision 64-bit IEEE 754 floating point.
- This data type is generally used as the default data type for decimal values, generally the default choice.
- Double data type should never be used for precise values such as currency.
- Default value is 0.0d.
- Example: double d1 = 123.4

**boolean:**

- boolean data type represents one bit of information.

- There are only two possible values: true and false.
- This data type is used for simple flags that track true/false conditions.
- Default value is false.
- Example: boolean one = true

#### **char:**

- char data type is a single 16-bit Unicode character.
- Minimum value is '\u0000' (or 0).
- Maximum value is '\uffff' (or 65,535 inclusive).
- Char data type is used to store any character.
- Example: char letterA ='A'

#### **Reference Data Types:**

- Reference variables are created using defined constructors of the classes. They are used to access objects. These variables are declared to be of a specific type that cannot be changed. For example, Employee, Puppy etc.
- Class objects, and various type of array variables come under reference data type.
- Default value of any reference variable is null.
- A reference variable can be used to refer to any object of the declared type or any compatible type.
- Example: Animal animal = new Animal("giraffe");

#### **TYPE CONVERSION**

Java supports two types of castings – **primitive data type casting** and **reference type casting**. Reference type casting is nothing but assigning one Java object to another object. It comes with very strict rules and is explained clearly in Object Casting. Now let us go for data type casting. Java data type casting comes with 3 flavors.

##### **1. Implicit casting**

##### **2. Explicit casting**

#### **1. Implicit casting (widening conversion)**

**A data type of lower size (occupying less memory) is assigned to a data type of higher size.**

This is done implicitly by the JVM. The lower size is widened to higher size. This is also named as **automatic type conversion**.

Examples:

```
int x = 10;           // occupies 4 bytes
double y = x;        // occupies 8 bytes
System.out.println(y); // prints 10.0
```

In the above code 4 bytes integer value is assigned to 8 bytes double value.

#### **2. Explicit casting (narrowing conversion)**

A data type of higher size (occupying more memory) cannot be assigned to a data type of lower size. This is not done implicitly by the JVM and requires **explicit casting**; a casting operation to be performed by the programmer. The higher size is narrowed to lower size.

```
double x = 10.5;     // 8 bytes
int y = x;           // 4 bytes ; raises compilation error
```

In the above code, 8 bytes double value is narrowed to 4 bytes int value. It raises error. Let us explicitly type cast it.

```
double x = 10.5;
int y = (int) x;
```

## Flow of Control

### PROGRAMMING CONSTRUCT

1. SEQUENCE
2. SELECTION
3. ITERATION

### Decision Making Statements ( Selection Construct) :-

There are two types of decision making statements in Java. They are:

- if statements
- switch statements

#### The if Statement:

An if statement consists of a Boolean expression followed by one or more statements.

Syntax:

The syntax of an if statement is:

```
if(Boolean_expression)
{
    //Statements will execute if the Boolean expression is true
}
```

If the Boolean expression evaluates to true then the block of code inside the if statement will be executed. If not the first set of code after the end of the if statement (after the closing curly brace) will be executed.

Example:

```
int x = 10;
if( x < 20 ){
    System.out.print("This is if statement");
}
```

This would produce the following result:

This is if statement

#### The if...else Statement:

An if statement can be followed by an optional else statement, which executes when the Boolean expression is false.

Syntax:

The syntax of an if...else is:

```
if(Boolean_expression){
    //Executes when the Boolean expression is true
}else{
    //Executes when the Boolean expression is false
}
```

Example:

```
int x = 30;

if( x < 20 ){
    System.out.print("This is if statement");
}else{
    System.out.print("This is else statement");
}
```

This would produce the following result:

This is else statement

### The if...else if...else Statement:

An if statement can be followed by an optional else if...else statement, which is very useful to test various conditions using single if...else if statement.

When using if , else if , else statements there are few points to keep in mind.

- An if can have zero or one else's and it must come after any else if's.
- An if can have zero to many else if's and they must come before the else.
- Once an else if succeeds, none of the remaining else if's or else's will be tested.

Syntax:

The syntax of an if...else is:

```
if(Boolean_expression 1){
    //Executes when the Boolean expression 1 is true
}else if(Boolean_expression 2){
    //Executes when the Boolean expression 2 is true
}else if(Boolean_expression 3){
    //Executes when the Boolean expression 3 is true
}else {
    //Executes when the none of the above condition is true.
}
```

Example:

```
int x = 30;

if( x == 10 ){
    System.out.print("Value of X is 10");
}else if( x == 20 ){
    System.out.print("Value of X is 20");
}else if( x == 30 ){
    System.out.print("Value of X is 30");
}else{
    System.out.print("This is else statement");
}
```

This would produce the following result:

Value of X is 30



### **Nested if...else Statement:**

It is always legal to nest if-else statements which means you can use one if or else if statement inside another if or else if statement.

Syntax:- The syntax for a nested if...else is as follows:

```
if(Boolean_expression 1){
    //Executes when the Boolean expression 1 is true
    if(Boolean_expression 2){
        //Executes when the Boolean expression 2 is true
    }
}
```

You can nest else if...else in the similar way as we have nested if statement.

Example:

```
int x = 30;
int y = 10;

if( x == 30 ){
    if( y == 10 ){
        System.out.print("X = 30 and Y = 10");
    }
}
```

This would produce the following result:

X = 30 and Y = 10

### **The switch Statement:**

A switch statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case.

Syntax:

The syntax of enhanced for loop is:

```
switch(expression){
    case value :
        //Statements
        break; //optional
    case value :
        //Statements
        break; //optional
    //You can have any number of case statements.
    default : //Optional
        //Statements
}
```

The following rules apply to a switch statement:

- The variable used in a switch statement can only be a byte, short, int, or char.
- You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.

- The value for a case must be the same data type as the variable in the switch and it must be a constant or a literal.
- When the variable being switched on is equal to a case, the statements following that case will execute until a break statement is reached.
- When a break statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.
- Not every case needs to contain a break. If no break appears, the flow of control will fall through to subsequent cases until a break is reached.
- A switch statement can have an optional default case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No break is needed in the default case.

Example:

```

char grade = 'C';

switch(grade)
{
    case 'A' :
        System.out.println("Excellent!");
        break;
    case 'B' :
    case 'C' :
        System.out.println("Well done");
        break;
    case 'D' :
        System.out.println("You passed");
    case 'F' :
        System.out.println("Better try again");
        break;
    default :
        System.out.println("Invalid grade");
}
System.out.println("Your grade is " + grade);
}

```

Compile and run above program using various command line arguments. This would produce the following result:

```

Well done
Your grade is a C

```

## Iteration Construct :-

### 1. Java while Loop

The while loop or while statement continually executes a block of statements while a particular condition is true. The while syntax can be written as:

```
while (expression) {  
    statement(s)  
}
```

The while loop evaluates expression, which must return a boolean value. If the while loop expression returns true, then the statements within the while block will be executed. The while loop continuously executes the statements within the block, until the expression returns false.

Here is a simple while loop example, which executes until i value became 10:

```
int i=0;  
while(i < 10){  
    //this block will executed until  
    //i value became 10  
    System.out.print(i+" ");  
    i=i+1;  
}
```

Output:

0 1 2 3 4 5 6 7 8 9

## 2. Java do-while Loop :-

A do-while loop is similar to a while loop, except that a do-while loop is guaranteed to execute at least one time. The difference between do-while and while loop is that do-while evaluates its condition at the bottom of the loop instead of the top. Therefore, the statements within the do block are always executed at least once.

Here is the syntax to write do-while loop:

```
do {  
    statement(s)  
} while (expression);
```

The do-while statement evaluates expression, which must return a boolean value. If the expression is true, the flow of control goes back to the do, and the statements within the loop executes again. This process repeats until the expression returns false.

Here is a simple do-while example:

```
int i = 0;  
do {  
    System.out.print(i+" ");  
    i=i+1;  
} while(i<10);
```

## 3. Java for Loop :-

The for statement or for loop provides a way to iterate over a range or list of values. Using for loop you can repeatedly loops until a particular condition is satisfied. The general form of the for statement can be expressed as follows:

```
for (initialization; condition for terminating; loop increment) {  
    statement(s)  
}
```

In the above statement, initialization expression initializes the loop; it is executed only once, as the loop begins.

When the termination condition returns false, then the loop terminates.

The increment expression is invoked after each iteration through the loop. Here you can either increment or decrement a value.

Here is a simple for loop example to display numbers from 1 to 10.

```
for(int i=1;i<=10;i++){
    System.out.print(i+" ");
}
System.out.println();
/**
 * another example to increment by 2 steps
 */
for(int i=1;i<=10;i=i+2){
    System.out.print(i+" ");
}
System.out.println();
/**
 * Below loop prints the numbers in reverse order
 */
for(int i=10;i>0;i--){
    System.out.print(i+" ");
}
System.out.println();
```

#### Output:

```
1 2 3 4 5 6 7 8 9 10
1 3 5 7 9
10 9 8 7 6 5 4 3 2 1
```

### Statement for Jump:--

#### 1. break statement in java:-

The break statement is one of the control statement in java. The break statement is generally used to break the loop before the completion of the loop. The break statement is used when we want to jump out of a single loop or single case in switch statement.

Here is a simple example for break statement:

```
for(int i=0;i<10;i++){
    if(i==5){
        System.out.println("breaking the for loop...");
        break;
    }
    System.out.println(i);
}
```

Output:

```
0
1
2
3
4
breaking the for loop...
```

## 2. continue statement in java:-

The continue statement skips the current iteration of a for, while, or do-while loop. The moment it encounters continue statement, it skips the rest of the statements in the loop and evaluates the expression for next iterations. The difference between break and continue statements are, break statement comes out of the loop, continue statement skips the current iteration and jumps to the next iteration in a single loop.

Here is a simple example for continue statement:

```
for(int i=1;i<=10;i++){
    /**
     * here this loop prints only odd numbers.
     * if below condition returns true, the current
     * iteration will be skipped.
     */
    if(i%2 == 0){
        continue;
    }
    System.out.println(i);
}
```

Output:

```
1
3
5
7
9
```

## Output Finding Questions:

Q.1. Write the output of the following code:

```
int x,y=0;
for(x=1;x<=5;++x)
y=x++;
--y
```

Ans. x=7,y=4

Q.2. Write the output of the following code

```
int f=1,i=2;
do{
f*=i;
}while(++i<5);
System.out.println(f);
```

Ans. f= 24

Q.3. What will be the value of j and k after execution of the following code:

```
int j=10,k=12;
if(k>=j)
{
k=j;
j=k;
}
```

Ans. j=10,k=10

Q.4. How many times ,the following loop gets executed?

```
i=0;
while(i>20)
{
//statements
}
```

Ans: zero times

Q,5. How many times ,the following loop gets executed?

```
i=0;
do{
//statements
}while(i>20);
```

Ans. one time

Q.6. What will be the contents of TextField after executing the following statement:

```
int num=4;
num=num+1;
if(num>5)
jTextField1.setText(Integer.toString(num));
else
jTextField1.setText(Integer.toString(num*4));
```

Ans. 20

Q.7. find the output

```
int number1=7,number2=8;
int second=73;
if(number1>0||number2>5)
if(number1>7)
jTextField1.setText("code worked");
else
jTextField1.setText("code might work");
else
jTextField1.setText("code will not work
might work");
```

Ans. code

Q.8. How many times will the following loop get executed?

```
x=5; y=36;
while(x<=y)
{
x+=6;
}
```

Q.9. What will be the content of the JTextArea1 after executing the following code?

```
int num=1;
do
{
jTextArea1.setText(Integer.toString(++num)+"\n");
num=num+1;
}while(num<=10);
```

Ans. 10

Q.10. Give the output for the following code fragment:

```
v=20;
do{
jOptionPane.showMessageDialog(numm,v+"");
}while(v<50);
jOptionPane.showMessageDialog(numm,v+"");
loop
```

Ans. infinite

Q.11. Give the value after executing following java code. Also find how many times the following loop will execute?

```
int a=10;
int b=12;
int x=5;
int y=6;
while(a<=b)
{
if(a%2==)
x=x+y;
else
x=x-y;
a=a+1;
}
```

Ans. x=11

Q.12. What will be the output produced by following code fragment?

```
float x=9;
float y=5;
int z=(int)(x/y);
switch(z)
{
case 1:x=x+2;
case 2: x=x+3;
default: x=x+1;
}
System.out.println("value of x:"+x);
```

Ans. 15

Q.13. Predict the output of the following code fragment:

```
int i,n;
n=0;i=1;
do
{
n++;i++;
}while(i<=5);
```

Ans. 6

Q.13. What will be the values of x and y after executing the following expressions.

```
int x=20,y=35;
x= y++ + x++;
y= ++y + ++x;
```

Ans x=56, y=93

Q.14. What will be the values of x and y after executing the following expressions.

```
int y=-1;
System.out.println("x:"+(-y--));
System.out.println("y:"+y);
```

Ans. x:= -1 ,y=-2

Q.15. What will be the values of x and y after executing the following expressions.

```
int x = 45;
x = x + x++;
System.out.println(x);
int y = 45;
y = y++ + y;
System.out.println(y);
```

Ans .x=90 , y=91

Q.16. What values will be assigned to the variable ua,ub,uc and fail after execution of the following program segment:

```
int i=0,ua=0,ub=0,uc=0,fail=0;
while(i<=5){
switch(i++){
{
case 1:++ua;
case 2:++ub;uc++;break;
```



```

case 3;
case 4:++uc;ua++;ub++;break;
default:++fail;
}

```

Ans. ua=1, ub=1, uc=0

Q.17. Predict an output of the following code fragment:

```

int i=1,j=0,n=0;
while(i<4)
{
for(j=1;j<=I;j++)
{
n+=1;
i=i+1;
}
System.out.println(n);
}

```

Ans. 6

Q..18. Give the output of the following code:

```

int m=100;
while(m>0)
{
if(m<10)
break;
m=m-10;
}
System.out.println("m is "+m);

```

Ans. 0

### **Error Finding Questions:**

Q.1. The following code has some errors. Rewrite the corrected code:

```

int i=2,j=5;
while(j>i)
{jTextField1.getText("j is greater");
j--;++ I;
}
JOptionPane.showMessageDialog("Hello");

```

Q.2. Identify the errors:

```

switch(ch)
{
case 'a':
case 'A':
case 'e':
case 'E':
case 'i':
case 'E':
case 'u':
case 'U':++vowels;
break;
default:++others;
}

```

Ans: Two case constant doesn't have the same value

```
Q.3. int j=5;
i==j+5;
if(i=j)
jTextField1.setText("I and j are equal");
else
jTextField1.setText(" I and j are unequal");
```

```
Ans. int i,j=5;
i=j+5;
if(i==j)
jTextField.setText("I and j are equal");
else
jTextField1.setText(" I and j are unequal");
```

Q.4. Rewrite the code after making correction. Underline the correction.

```
Int sum,value;inct;
Int i
For(i==0;i<=10;i++)
Sum=sum+I;
Inct++;
Ans.
```

```
int sum,value,inct;
int i;
for(i=0;i<=10;i++)
sum=sum+i;
inct++;
```

Q.5. The following code has some error(s). Rewrite the correct code underlining all the corrections made.

```
Int y=3;
Switch(y)
{
case 1: System.out.println("yes its one");
case >2: System.out.println("yes it is more than two");
break;
case else: System.out.println("invalid number");
```

```
Ans. int y=3;
switch(y)
{
case 1 : System.out.println("yes its one");
break;
case 2: System.out.println("yes its more than two");
break;
default: System.out.println("invalid number");
}
```

Q.6. Rewrite the following java code after underlining the correction made:

```
int x==0;
int n=Integer.parseInt(jLabel1.getText( ));
```

Ans. `int x=0;`  
`int n=Integer.parseInt(jLabel1.getText( ));`

Q.7. find out errors if any:

```
m=1;n=0;
For(;m+n<19;++n)
System.out.println("hello");
M=m+10;
```

Ans. `m=1;n=0;`  
`for(;m+n<19;++n)`  
`System.out.println("hello");`  
`m=m+10;`

Q.8. The following code has some error(s). Rewrite the correct code underlining all the corrections made.

```
int y=6,p;
Do
{
y=3.14*y;
p=y%10;
If p=2
System.out.print("two");
While(y>1)
Ans.int y=6,p;
do
{
y=3.14*y;
p=y%10;
if( p==2)
System.out.print("two");
}while(y>1);
```

### Rewrite Questions

Q.1. Rewrite the following program code using a for loop:

```
int I,sum=0;
while(I<10)
{
sum+= I ;
I+=2;
}
```

Ans.  
`int I,sum=0;`  
`for(I=0;I<10; I+=2)`  
`{`  
`sum+=I;`  
`}`

Q.2. rewrite the following code using while loop:

```
int i,j;
for(i=1;i<=4;i++)
{
for(j=1;j<=I;++j)
System.out.print(j);
}
System.out.println( );
}
```

Ans.

```
int i=1,j;
while(i<=4)
{
J=1;
while(j<=i)
{
System.out.print(j);
++j;
}
i++;
System.out.println( );
}
```

Q.3. Write an equivalent while loop for the following code:

```
int sz=25;
for(int i=0,sum=0;i<sz;i++)
sum+=I;
System.out.println(sum);
```

Ans. int sz=25;

```
int i=0,sum=0;
```

```
while(i<sz)
```

```
{
sum+=i;
```

```
i++;
```

```
}
```

```
System.out.println(sum);
```

Q.4. Rewrite the following if-else segment using switch –case statement

```
char ch='A';
```

```
if(ch=='A')
```

```
System.out.println("Account");
```

```
if((ch=='C')||(ch=='G'))
```

```
System.out.println("Admin");
```

```
if(ch=='F')
```

```
System.out.println("advisor");
```

Ans.

```
char ch='A';
switch(ch)
{
case 'A':System.out.println("account");
break;
case 'C':
case 'G': System.out.println("admin");
break;
case 'F': System.out.println("advisor");
}
```

Q..5. Rewrite the following code using while loop

```
int i, j;
for(i=1,j=2;i<=6;i++,j+=2)
System.out.println(i++);
System.out.println("finished!!!");
```

Ans.

```
int i=1,j=2;
while (i<=6)
{
System.out.println(i++);
i++;
j+=2;
}
System.out.println("finished!!!");
```

Q.6. Rewrite the following code using for loop.

```
int i=0;
while(++i<20)
{
if(i==8)
break;
System.out.println(i++);
}
```

Ans.

```
int i;
for(i=1;i<20;++i)
{ if(i==8)
break;
System.out.println(i++);
}
```

Q.7. Rewrite the code using switch statement:

```
if(k==1)
day="Monday";
else if(k==2)
day="Tuesday";
else if (k==3)
day="Wednesday";
else day="-";
}
```

Ans. switch(k)

```
{
case 1:day="Monday";
break;
case 2:day="Tuesday";
break;
case 3 :day="Wednesday";
break;
default: day= " ";
}
```

### Questions

1. How do you write an infinite loop using the for statement?
2. How do you write an infinite loop using the while statement?

3. What will be the output of :  
if (aNumber >= 0)  
if (aNumber == 0)  
System.out.println("first string");  
else System.out.println("second string");  
System.out.println("third string");  
If aNumber is (i) 0 (ii) 2

4.. What will be the output of the program?

```
int count = 1;
do
{
System.out.println("Count is: " + count);
count++;
} while (count < 11);
```

5. What will be the output of the program

```
int i=1,j=-1;
switch(i)
{
case 1,-1: j=1;
case 2 : j=2;
default :j=0;
}
```

6. . What will be the output of the program?

```
int i = 1, j = 10;
do
{
    if(i > j)
    {
        break;
    }
    j--;
} while (++i < 5);
System.out.println("i = " + i + " and j = " + j);
```

7. . What will be the output of the program?

```
final static short x = 2;

for (int z=0; z < 3; z++)
{
    switch (z)
    {
        case x: System.out.print("0 ");
        case x-1: System.out.print("1 ");
        case x-2: System.out.print("2 ");
    }
}
```

8. What will be the output of the program?

```
for (int i = 0; i < 4; i += 2)
{
    System.out.print(i + " ");
}
System.out.println(i);
```

9. What will be the output of the program?

```
int x = 3;
int y = 1;
if (x = y)
{
```

```
System.out.println("x = " + x);  
}
```

10. What will be the output of the program?

```
for(int i = 0; i < 3; i++)  
{  
    switch(i)  
    {  
        case 0: break;  
        case 1: System.out.print("one ");  
        case 2: System.out.print("two ");  
        case 3: System.out.print("three ");  
    }  
}  
System.out.println("done");
```

11. What will be the output of the program?

```
int i = 1, j = 0;  
switch(i)  
{  
    case 2: j += 6;  
    case 4: j += 1;  
    default: j += 2;  
    case 0: j += 4;  
}  
System.out.println("j = " + j);
```

12. What will be the output of the program?

```
int i = 1, j = 0;  
switch(i)  
{  
    case 2: j += 6;  
    case 4: j += 1;  
    default: j += 2;  
    case 0: j += 4;  
}  
System.out.println("j = " + j); }
```

13. What will be the output of the program?

```
long a=78345,s1=0,s2=0,r;  
while(a>0)  
{  
    r=a%10;  
    if(r%4==0)  
        s1+=r;  
    else s2+= r;
```



```
a/=10;
}
System.out.println("S1="+s1);
System.out.println("S2="+s2);
```

14. What will be the output of the program?

```
int nos=100;
while(nos>=45)
{
    if(nos%5==0)
        nos+=10;
    else
        nos+=20;
}
```

15. What will be the output of the:

```
(ii) byte b;
double d=417.35;
b=(byte)d;
System.out.println(b);
```

```
(iii) int m=100;
int n=300;
while(++m< --n)
System.out.println(m+" "+n);
```

```
(iv) int x=10,y=20;
if((x<=y)||((x==5)>10))
System.out.println(x);
else
System.out.println(y);
```

```
(v) int x=10;
float y=10.0;
System.out.println((x>y)?true:false);
```

## JAVA GUI PROGRAMMING USING SWING COMPONENTS

- **JPanel** is Swing's version of the AWT class Panel and uses the same default layout, Flow Layout. JPanel is descended directly from JComponent.
- **JFrame** is Swing's version of Frame and is descended directly from that class. The components added to the frame are referred to as its contents; these are managed by the content Pane. To add a component to a JFrame, we must use its content Pane instead.
- **JInternalFrame** is confined to a visible area of a container it is placed in. It can be iconified, maximized and layered.
- **JWindow** is Swing's version of Window and is descended directly from that class. Like Window, it uses Border Layout by default.
- **JDialog** is Swing's version of Dialog and is descended directly from that class. Like Dialog, it uses Border Layout by default. Like JFrame and JWindow, JDialog contains a root Pane hierarchy including a content Pane, and it allows layered and glass panes. All dialogs are modal, which means the current thread is blocked until user interaction with it has been completed. JDialog class is intended as the basis for creating custom dialogs; however, some of the most common dialogs are provided through static methods in the class JOptionPane.
- **JLabel**, descended from JComponent, is used to create text labels.
- The abstract class Abstract Button extends class JComponent and provides a foundation for a family of button classes, including **JButton**.
- **JTextField** allows editing of a single line of text. New features include the ability to justify the text left, right, or center, and to set the text's font.
- **JPasswordField** (a direct subclass of JTextField) you can suppress the display of input. Each character entered can be replaced by an echo character. This allows confidential input for passwords, for example. By default, the echo character is the asterisk, \*.
- **JTextArea** allows editing of multiple lines of text. JTextArea can be used in conjunction with class JScrollPane to achieve scrolling. The underlying **JScrollPane** can be forced to always or never have either the vertical or horizontal scrollbar; JButton is a component the user clicks to trigger a specific action.
- **JRadioButton** is similar to JCheckbox, except for the default icon for each class. A set of radio buttons can be associated as a group in which only one button at a time can be selected.
- **JCheckBox** is not a member of a checkbox group. A checkbox can be selected and deselected, and it also displays its current state.
- **JComboBox** is like a drop down box. You can click a drop-down arrow and select an option from a list. For example, when the component has focus, pressing a key that corresponds to the first character in some entry's name selects that entry. A vertical scrollbar is used for longer lists.
- **JList** provides a scrollable set of items from which one or more may be selected. JList can be populated from an Array or Vector. JList does not support scrolling directly, instead, the list must be associated with a scroll pane. The view port used by the scroll pane can also have a user-defined border. JList actions are handled using ListSelectionListener.

- Focus:- The control under execution is said to have the focus. The control having the focus obtains input from the user.
- `getText()`:- `getText()` method is used to obtain the text from a `(jTextField)` during the run time.
- `setText()`:- `setText()` method is used to set or change the text of a `jTextField` during run time.

### SWING controls methods and properties

These are the Swing Controls available with NetBeans IDE and their concern methods and properties are given below.

SWING CONTROLS	METHODS	PROPERTIES
<code>jButton</code>	<code>getText()</code> <code>setText()</code>	<ul style="list-style-type: none"> <li>• Background</li> <li>• Enabled</li> <li>• Font</li> <li>• Foreground</li> <li>• Text</li> <li>• Label</li> </ul>
<code>jLabel</code>	<code>getText()</code>	<ul style="list-style-type: none"> <li>• Background</li> <li>• Enabled</li> <li>• Font</li> <li>• Foreground</li> <li>• Text</li> </ul>
<code>jTextField</code>	<code>getText()</code> <code>isEditable</code> <code>isEnabled</code> <code>setText()</code>	<ul style="list-style-type: none"> <li>• Background</li> <li>• Enabled</li> <li>• Editable</li> <li>• Foreground</li> <li>• Text</li> </ul>
<code>jRadioButton</code>	<code>getText()</code> <code>setText()</code> <code>isSelected()</code> <code>setSelected()</code>	<ul style="list-style-type: none"> <li>• Background</li> <li>• Enabled</li> <li>• Font</li> <li>• Foreground</li> <li>• Buttongroup</li> <li>• selected</li> <li>• Label</li> </ul>
<code>jCheckBox</code>	<code>getText()</code> <code>setText()</code> <code>isSelected()</code> <code>setSelected()</code>	<ul style="list-style-type: none"> <li>• ButtonGroup</li> <li>• Font</li> <li>• Foreground</li> <li>• Label</li> <li>• Selected</li> <li>• Text</li> </ul>
<code>jComboBox</code>	<code>getSelectedItem()</code> <code>getSelectedIndex()</code> <code>setModel()</code>	<ul style="list-style-type: none"> <li>• Background</li> <li>• Buttongroup</li> <li>• Editable</li> <li>• Enabled</li> <li>• Font</li> </ul>

		<ul style="list-style-type: none"> <li>• Foreground</li> <li>• Model</li> <li>• SelectedIndex</li> <li>• SelectedItem</li> <li>• SelectionMode</li> <li>• Text</li> </ul>
jTable	addRow( ) getModel( )	<ul style="list-style-type: none"> <li>• Model</li> </ul>
jOptionPane	showMesageDialog( )	<ul style="list-style-type: none"> <li>• getRowCount( )</li> <li>• removeRow( )</li> <li>• addRow( )</li> </ul>

### Some Important Questions with Answers

1. Which window is used to design the form

Ans. Design Window

2. Which window contains the Swing Controls components?

Ans. Palette window

3. What is the most suitable component to accept multiline text.

Ans. TextArea

4. Name the different list types controls offered by Java Swing.

Ans. (i) JList (ii) JComboBox

5. Name any two commonly used method of List.

Ans. (i) getSelectedIndex( ) (ii) getSelectedItem( )

6. By default a combo box does not offer editing feature. How would you make a combo box editable.

Ans. By setting its editable property to true.

7. What is the name of event listener interface for action events?

Ans. ActionListener

8. What is the difference between a Container and Component control?

Ans. A container is a control that can hold other controls within it. E.g. Panel (there can be multiple controls inside Panel, Frame (where you can put so many controls on it.)

Component: Controls inside the container are known as components

9. Differentiate between

- (a) TextField and TextArea components
- (b) TextField and PasswordField
- (c) Combo box and list box

(a) Ans. The TextField allows the user to enter a single line of text only. But TextArea component allows to accept multiline input from the user or display multiple lines of information.

(b) TextField displays the obtained text in unencrypted form whereas password field displays the obtained text in encrypted form. This component allows confidential input like passwords

(c) List box :- In the list box, the users can only select from the list of choices, but in combobox a user can select from the list of choices as well as enter his/her choice (combobox=listbox+textfield)

10. Why are data types important?

Ans. Data Types define the way the values are stored, the range of the values and the operations that are associated with that type.

11. What is a variable?

Ans. Variable is named temporary storage locations.

12. What is an identifier?

Ans. Identifiers are fundamental building block of a program and are used as the general terminology for the names given to different parts of the program .

13. What is casting? When do we need it?

Ans. Assigning a value of one type to the variable of another type is known as type casting

```
int x=10;
```

```
byte y= (byte) x;
```

14. Is Java case sensitive? What is meant by case sensitive?

Ans. Yes, java is case sensitive. Case sensitive means upper case and lower case letters are treated differently.

15. What does getPassword( ) on a password field return?

Ans. A character array.

16. Which component is the best suited to accept the country of the user?

Ans.. ListBox and ComboBox both

17. What command do you need to write in actionPerformed( ) event handler of a button I order to make it Exit button?

Ans. System.exit(0);

18. Which control displays text that the user cannot directly change or edit?

Ans. Label

19. Which control provides basic text editing facility?

Ans. TextField

20. Occurrence of an activity is called.

Ans. Event

21. Which property is used to set the text of the label?

Ans. Text

22. The object containing the data to be exhibited by the combo box by which property

Ans. Model

23. What is GUI programming?

Ans. We can create a GUI application on Java platform using Swing API ,which is part of java foundation classes (JFC).

24. What is an event? What is event handler, source, object?

Ans. An event is occurrence of some activities either initiated by user or by the system. In order to react, you need to implement some event handling system in your application.

Event source:-It is the GUI component that generates the event eg. Button

Event Handler or Event Listener:- It is implemented as in the form of code .It receives and handles events through listener interface.

Event object or message:- It is created when event occurs. It contains all the information about the event which includes source of event and type of even etc.

25. Which property would you set the setting the password character as \$?

Ans. echoChar

26. Which method returns the password entered in a password field?

Ans. getPassword( )

27. Which method would you determine the index of selected item in a list?

Ans. getSelectedIndex(int Index)

28. Which method would you use to insert an item at specified index, in the list?

Ans. setSelectedIndex(4)

29. How you can determine whether 5<sup>th</sup> item in a list is selected or not?

Ans. isSelectedIndex(4)

30. Which method you would use to insert "hello" at 10<sup>th</sup> position in the TextArea control.

Ans. insert("hello",9) //index starts from zero

31. Which property would you like to set to make a combo box editable?

Ans. Editable

32. What do you understand by focus.

Ans. A Focus is the ability to receive user input/response through Mouse/Keyboard .When object or control has focus, it can receive input from user.

a) An object or control can receive focus only if its enabled and visible property are set to true.

b) Most of the controls provide FOCUS\_GAINED( ) and FOCUS\_LOST( ) method

33. What is meant by scope of a variable ?

Ans. In java ,a variable can be declared anywhere in the program but before using them.

1.The area of program within which a variable is accessible ,known as its scope.

2.A variable can be accessed within the block where it is declared

```
{
int x=10;
if(a>b)
{
int y=5;
//scope of x and y
}
else
{
int z=3;
//scope of z
}
.....
}
```


## DESIGN PROBLEMS

Q1.

Glamour Garments has developed a GUI application for their company as shown below :

The company accepts payments in 3 modes- cheque , cash and credit cards. The discount given as per mode of payment is as follows.

Mode of Payment	Discount
Cash	8%
Cheque	7%
Credit Card	Nil



If the Bill Amount is more than 15000 then the customer gets an additional discount of 10% on Bill Amount.

- (1) Make Discount and Net amount uneditable.
- (2) Write codes for calculate Discount and calculate Net Amount Buttons
- (3) Write code to exit program when STOP button is clicked.

### Solution:

(1)

```
txtDiscount.setEditable(false);
```

```
txtNetAmt.setEditable(false);
```

(2)

```
//Code for calculate button
```

```
String name= txtname.getText( );
```

```
double bm=Double.parseDouble(txtbillamt.getText( ));
```

```
double disc=0.0, netAmt=0.0;
```

```
String s= cmbMode.getSelectedItem( );
```

```
if(s.equals("Cash"))
```

```
{
```

```
    disc= 0.08*bm;
```

```
}
```

```
else if(s.equals("Cheque"))
```

```
{
```

```
    disc=0.07*bm;
```

```
}
```

```

else if(s.equals("Cash"))
{
    disc=0;
}
netAmt=bm-disc; txtDiscount.setText(" "+disc); txtNetAmt.setText(" "+netAmt);
(3)
//code for stop button

System.exit(0);

```

2. Create a Java Desktop Application to find the incentive (%) of Sales for a Sales Person on the basis of following feedbacks:

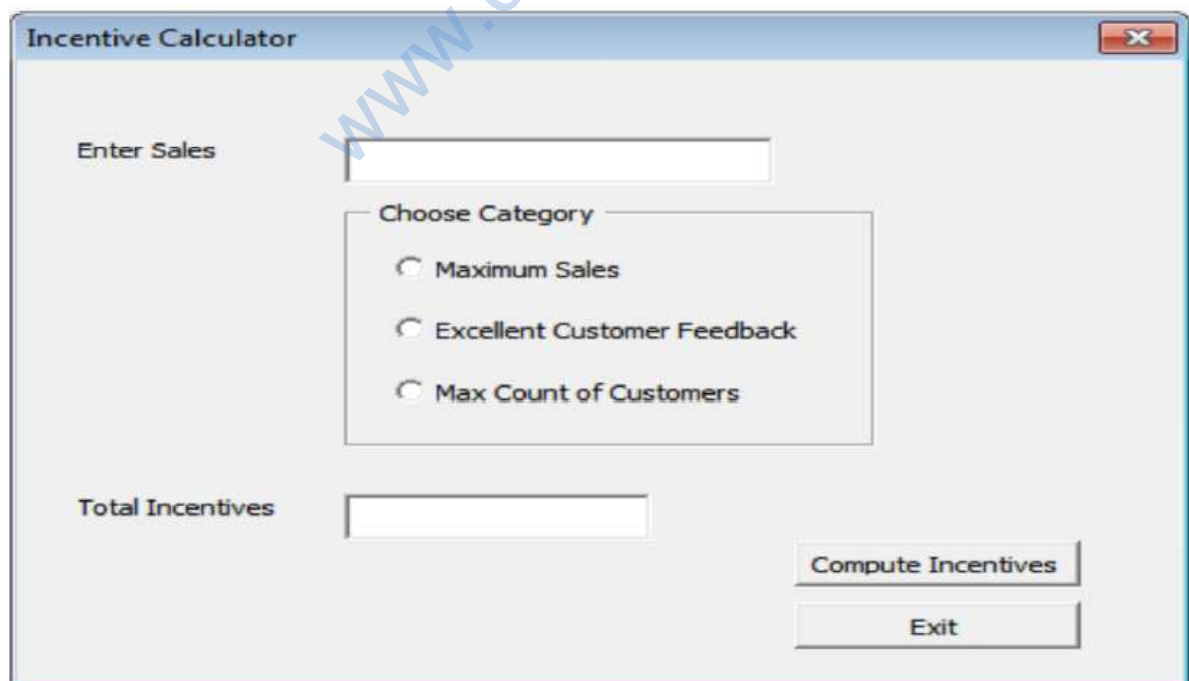
Feedback	Incentive (%)
Maximum Sales	10
Excellent Customer Feedback	8
Maximum Count Customer	5

Note: that the sales entry should not be space. Calculate the total incentive as :  
Sales amount\* Incentive.

The feedback will be implemented in JCheckBox controls. Using a JButton's (Compute Incentive)

click event handler, display the total incentives in a JTextField control. Assume the nomenclature of the swing components of your own.

Note that the JFrame from IDE window will be shown as given:





Ans:- private void btnIncActionPerformed (java.awt.ActionEvent evt)

```
{
    int sales = 0;
    if (!txtSales.getText().trim().equals(""))
    {
        sales=Integer.parseInt(txtSales.getText().trim ());
    }
    double incentive = 0.0;
    if (jCheckBox1.isSelected ( ))
    {
        incentive = incentive + 0.1;
    }
    if (jCheckBox2.isSelected ( ))
    {
        incentive = incentive + 0.8;
    }
    if (jCheckBox3.isSelected ( ))
    {
        incentive = incentive + 0.05;
    }
    txtInc.setText ( "" + Math.round(sales * incentive));
}
```

3. Assume the following interface built using Netbeans used for bill calculation of a ice-cream parlor. The parlor offers three varieties of ice-cream – vanilla, strawberry, chocolate. Vanilla ice-cream costs Rs. 30, Strawberry Rs. 35 and Chocolate Rs. 50. A customer can chose one or more ice-creams, with quantities more than one for each of the variety chosen. To calculate the bill parlor manager selects the appropriate check boxes according to the varieties of ice-cream chosen by the customer and enter their respective quantities.

Write Java code for the following:

- On the click event of the button 'Calculate', the application finds and displays the total bill of the customer. It first displays the rate of various ice-creams in the respective text fields. If a user doesn't select a check box, the respective ice-cream rate must become zero. The bill is calculated by multiplying the various quantities with their respective rate and later adding them all.
- On the Click event of the clear button all the text fields and the check boxes get cleared.
- On the click event of the close button the application gets closed.



Ans: (a)

```
private void jButtonCalculateActionPerformed(java.awt.event.ActionEvent evt)
{
    if(jchkStrawberry.isSelected() == true)
        jTxtPriceStrawberry.setText("35");
    else
    {
        jTxtPriceStrawberry.setText("0");
        jTxtQtyStrawberry.setText("0");
    }
    if(jchkChocolate.isSelected() == true)
        jTxtPriceChocolate.setText("50");
    else
    {
        jTxtPriceChocolate.setText("0");
        jTxtQtyChocolate.setText("0");
    }
    if(jchkVinella.isSelected() == true)
        jtxtPriceVinella.setText("30");
    else
    {
        jtxtPriceVinella.setText("0");
        jTxtQtyVinella.setText("0");
    }
}
```

```

int r1,r2,r3,q1,q2,q3,a1,a2,a3,gt;
r1=Integer.parseInt(jTxtPriceStrawberry.getText( ));
r2=Integer.parseInt(jTxtPriceChocolate.getText( ));
r3=Integer.parseInt(jtxtPriceVinella.getText( ));
q1=Integer.parseInt(jTxtQtyStrawberry.getText( ));
q2=Integer.parseInt(jTxtQtyChocolate.getText( ));
q3=Integer.parseInt(jTxtQtyVinella.getText( ));
a1=r1*q1;
jTxtAmtStrawberry.setText(""+a1);
a2=r2*q2;
jTxtAmtChocolate.setText(""+a2);
a3=r3*q3;
jTxtAmtVinella.setText(""+a3);
gt=a1+a2+a3;
jTxtTotalAmt.setText(""+gt);
}

```

Ans.(b)

```

private void jBtnClearActionPerformed(java.awt.event.ActionEvent evt)
{
    jTxtPriceStrawberry.setText("");
    jTxtPriceChocolate.setText("");
    jtxtPriceVinella.setText("");
    jTxtQtyStrawberry.setText("");
    jTxtQtyChocolate.setText("");
    jTxtQtyVinella.setText("");
    jTxtAmtStrawberry.setText("");
    jTxtAmtChocolate.setText("");
    jTxtAmtVinella.setText("");
    jchkStrawberry.setSelected(false);
    jChkChocolate.setSelected(false);
    jChkVinella.setSelected(false);
}

```

Ans: (c)

```

private void jBtnCloseActionPerformed(java.awt.event.ActionEvent evt)
{
    System.exit(0);
}

```

4. Read the following case study and answer the questions that follow.

- TeachWell Public School wants to computerize the employee salary section.
- The School is having two categories of employees : Teaching and Non Teaching. The Teaching employees are further categorized into PGTs, TGTs and PRTs having different Basic salary.
- The School gives addition pay of 3000 for employees who are working for more than 10 years.

Employee Type	Basic Salary	DA (% of Basic Sal)	HRA (% of Basic Sal)	Deductions (% of Basic sal)
Non-Teaching	25001	31	30	12
PGT	14500	30	30	12
TGT	12500	21	30	12
PRT	11500	20	25	12

(a) Write the code to calculate the Basic salary, deductions, gross salary and net salary based on the given specification. Add 3000 to net salary if employee is working for more than 10 years.

Gross salary=Basic salary + DA + HRA

Net salary = Gross salary – deductions

(b) Write the code to exit the application.

(c) Write the code to disable textfields for gross salary, deductions and netsalary.

**Ans: (a)**

```
double bs=0,da=0,net=0,ded=0,gross=0,hra=0;
if (rdnon.isSelected() == true)
{
    bs=12500;
    da=(31*bs)/100;
    hra=(30*bs)/100;
```

```

        ded=(12*bs)/100;
    }
    else if (rdpgt.isSelected() == true)
    {
        bs=14500;
        da=(30*bs)/100;
        hra=(30*bs)/100;
        ded=(12*bs)/100;
    }
    else if (rdtgt.isSelected() == true)
    {
        bs=12500;
        da=(21*bs)/100;
        hra=(30*bs)/100;
        ded=(12*bs)/100;
    }
    else if (rdprt.isSelected() == true)
    {
        bs=11500;
        da=(20*bs)/100;
        hra=(25*bs)/100;
        ded=(12*bs)/100;
    }
    gross=bs+da+hra;
    net = gross - ded;
    if(chk10.isSelected() == true)
    {
        net=net+3000;
    }
    tfded.setText(""+ded);
    tfgross.setText(""+gross);
    tfnet.setText(""+net);
    tfbs.setText(""+bs);

```

Ans:(b)

```
System.exit(0);
```

Ans:(c)

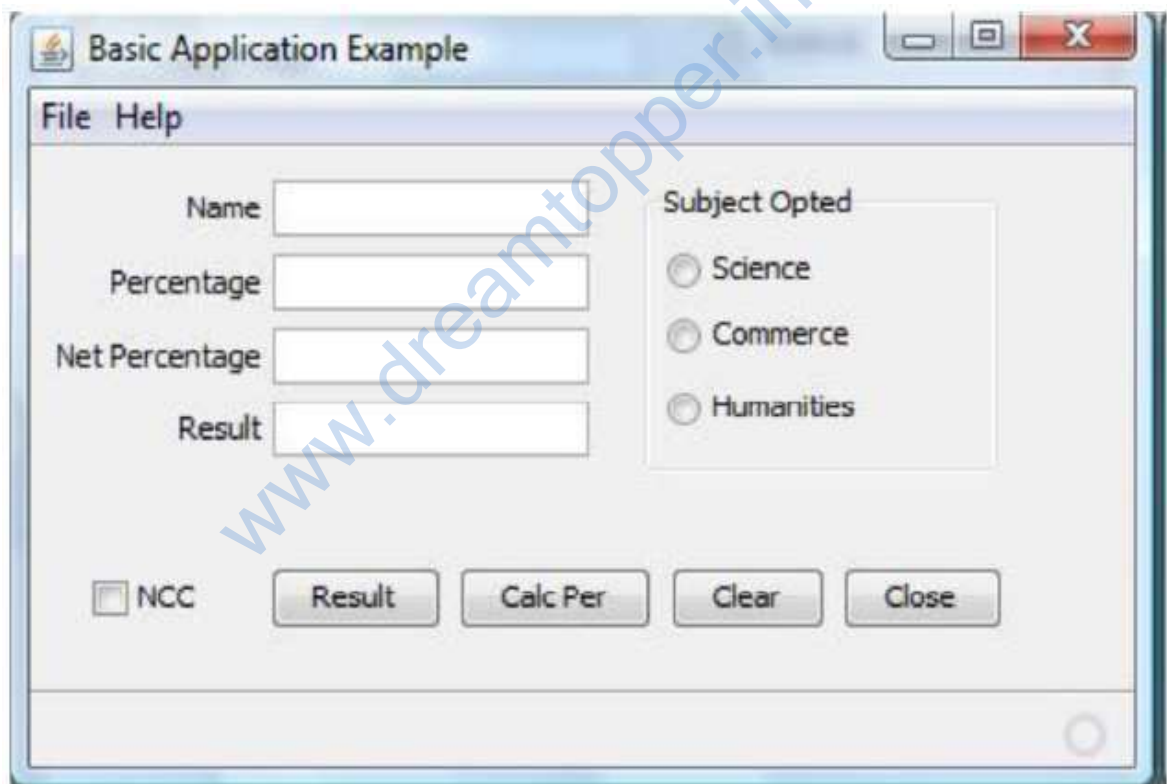
```

tfgross.setEditable(false);
tfded.setEditable(false);
tfnet.setEditable(false);

```

5. ABC School uses the following interface built in java to check the eligibility of a student for a particular stream from science, commerce and humanities. The user first enters the total percentage and selects the desired stream by selecting the appropriate option button. An additional 5% is marks is given to students of NCC. Write Java Code for the following

- On Action event of the button 'Calc Percentage' Net percentage of the student is calculated and displayed in the appropriate text filed. Net percentage is same as that of the actual percentage if the student doesn't opts for NCC otherwise 5% is added to actual percentage.
- On Action event of the button 'Result', the application checks the eligibility of the students. And display result in the appropriate text field. Minimum percentage for science is 70, 60 for commerce and 40 for humanities.
- On the Click event of the clear button all the text fields and the check boxes get cleared.
- On the click event of the close button the application gets closed.



Ans:

a.

```
private void jBtnCalcPerActionPerformed(java.awt.event.ActionEvent evt)
{
    int p;
    p=Integer.parseInt(jTextField2.getText( ));
```

```

if (jCheckBox1.isSelected( ))
    p=p+5;
    jTextField3.setText(Integer.toString(p));
}

```

b.

```

private void jBtnResultActionPerformed(java.awt.event.ActionEvent evt)
{
    int p;
    p=Integer.parseInt(jTextField3.getText( ));
    if( jRadioButton1.isSelected( ))
    {
        if ( p>=70)
            jTextField4.setText("Eligible for all subject");
        else
            jTextField4.setText("Not Eligible for science");
    }
    else if( jRadioButton2.isSelected( ))
    {
        if ( p>=60 )
            jTextField4.setText("Eligible for Commerce and Humanities");
        else
            jTextField4.setText("Not Eligible for Science and Commerce");
    }
    else
    {
        if ( p>=40 )
            jTextField4.setText("Eligible for Humanities");
        else
            jTextField4.setText("Not Eligible for any subject ");
    }
}

```

c.

```

private void jBtnClearActionPerformed(java.awt.event.ActionEvent evt)
{
    jTextField1.setText(" ") OR jTextField1.setText(null)
    jTextField1.setText(" ") OR jTextField1.setText(null)
    jTextField1.setText(" ") OR jTextField1.setText(null)
    jTextField1.setText(" ") OR jTextField1.setText(null)
}

```

```
        jCheckbox1.setSelected(false);
    }
```

d.

```
private void jBtnCloseActionPerformed(java.awt.event.ActionEvent evt)
{
    System.exit(0);
}
```

### Unsolved Questions:

1. Describe the relationship between properties, methods and events.
2. What is container tag?
3. What does a getPassword( ) method of a password field returns?
4. What will be the contents of JTextArea1 after executing the following statement: 1
5. JTextArea1.setText("Object\nOriented\tProgramming");
6. What is difference between JRadioButton and JCheckBox?
7. What is Layout Manager? Discuss briefly about layout managers offered by NetBeans?
8. Name three commonly used properties and methods of the following controls.
9. (a) text field (b) text area (c) label (d) Check Box (e) button.
10. What is dispose( ) used for ?
11. What is the difference between-
  - (a) Text field & Text area
  - (b) List & Combo
  - (c) Radio Button & Check Box
12. What is the significance of following properties of a text area?
  - (a) lineWrap (b) wrapStyleword
13. What is the significance of a button group ? How do you create a button group?
14. Discuss about some commonly used properties of lists and a combo boxes.
15. What methods obtain the current selection of a combo box? Give a code example.
16. The FOR U SHOP has computerized its billing. A new bill is generated for each customer. The shop allows three different payment modes. The discount is given based on the payment mode.



Credit Card Type	Shopping Amount	Discount
Cash	< 10000 20 %	20 %
	>= 10000 25 %	25%
Cheque	< 15000	10 %
	>= 15000	15 %
Credit Card	< 10000	10 %
	>= 10000	12%

- a) Write the code for the CmdClear Button to clear all the Text Fields.
- b) Write the code for the CmdCalc Button to display the Discount Amount and Net Price in the TxtDisc and the TxtNet Text Fields respectively.

## **PROGRAMMING GUIDELINES :**

### **Stylistic Guidelines:**

- Use of meaningful names for identifiers
- Ensure clarity of expressions
- Use of comments and indentation
- Insert blank lines and blank spaces

### **Characteristics for a good program:**

- Effective and efficient
- User friendly
- Self documenting code
- Reliable
- Portable

### **Stages of Program Development Process :**

A program development process is a step by step process where each stage contributes to building of an effective and efficient program.

Stages are as follows

- Crack the problem
- Code the algorithm
- Compile the program
- Execute the program

### **Types of Errors :**

- **Compile Time error**-occurs during compile time .When a program compiles its source code is checked for rules of programming language.

#### **Types of compile time error**

- (i) **Syntax error:** it occurs when a grammatical rule of java is violated. Formal set of rules defined for writing any statement in a language is known as syntax. Syntax errors occur when syntax rules of any programming language are violated. These errors occur during compilation of the application but Some of the common examples of syntax errors are missing semicolon, missing parenthesis and using incompatible data types
  - (ii) **Semantic error:** it occurs when statement are not meaningful
- **Run time error:** it occurs during the execution of the program. If an application is syntactically correct then it is compiled and translated into machine language and is ready to be executed. Run time errors occur during the execution of an application. These errors will result in abnormal termination of the application. Some common examples of runtime errors are Division by zero, trying to convert to number (int or double) from an empty
  - **Logical error:** it occurs due to wrong logic of a program.

## PROBLEM SOLVING METHODOLOGY AND TECHNIQUES

Steps to create a working program are:-

1. Understand the problem well
2. Analyze the problem to
  - a) Identify minimum number of inputs required for output.
  - b) Identify processing components.
3. Design the program by
  - a) Deciding step by step solution
  - b) Breaking down solution into simple steps
4. Code the program by
  - a) Identifying arithmetic and logical operation required for solution
  - b) Using appropriate control structure such as conditional or looping control
5. Test and Debug your program by
  - a) Finding error in it
  - b) Rectifying the error.

Virtually all applications have defects in them called 'bugs' and these need to be eliminated. Bugs can arise from errors in the logic of the program specification or errors in the programming code created by a programmer. Testing means the process of executing the application with possible set of input data in order to find probable errors. **Debugging** means correction of those errors in the application. In the testing and debugging stage, we should try out all possible inputs in order to make our application error free.

6. Complete your documentation :- Documentation means the instructions and information about the usage of the application. Providing the documentation makes it easier for the end user to understand the functionality of the application.

For example, giving appropriate comments in all our applications is part of documentation as it clearly tells the user and the programmer about what a particular part of the code is doing

7. Maintain your program:- The maintenance involves the rectification of previously undetected errors and changes that are to be made for the enhancement of the functionality of the application. An updated version of the software with the reported bugs corrected and enhancements is then sent as a replacement to the end user.

### Questions and Answer on Programming Guidelines

Q.1. Excessive use of comments increases the execution time of your program(True/False)  
Justify your answer.

Ans. No, comments don't add time to program execution .As comments are only for documentation purpose and they are non executable statement.(ignored by the compiler).

Q.2. differentiate between compile time and run time errors.

Ans. Compile time errors occur due to violation of grammatical rules of a programming language. Run time errors occur during execution of program.

Compile time errors are easy to correct as we the get error message corresponding to that which give an idea to correct it. Run time errors causes abnormal termination of program.

e.g. Compile time error  $A==B+C$

Run time error divide by zero error

**Q.3. Which error is harder to locate and why ?**

**Ans** logical errors are harder to locate . logical errors occur due to errors in the logic of a program . When a program is syntactically correct, even running properly but not giving a desired output, it means that it has a logical error.

One common example of logical error is when we write a statement  $Eng + Math + Gk / 3$  instead of  $(Eng + Math + Gk) / 3$  to calculate average of marks of 3 subjects.

**Q.4. Explain the following terms:**

- a) Exception handling  
A run time error is called an exception, which causes abnormal termination of program. To handle such type of errors/exception is called exception handling. In java exception handling is done by try { } block. Statements that can raise exception are put in try { } block and its handling code is written in catch { } block
- b) Syntax:  
Formal set of rules defined for writing any statement in language is known as syntax. Example –Every line in JAVA should be terminated by semicolon (;)
- c) Portability – portability means an application should run on different platform without doing any changes
- d) Pretty printing :Pretty printing is the formatting of a program to make it more readable. These formatting conventions usually consist of changes in positioning, spacing, color, contrast, size and similar modifications intended to make the content easier to view, read and understand. Pretty printers for programming language source code are sometimes called code beautifiers or syntax highlighters .net beans supports pretty printing and the shortcut key to format any source code in net beans is Alt+Shift+F.
- e) Syntax error: Formal set of rules define for writing any statement in a language is known as syntax. Syntax errors occur when syntax rules of any programming languages are violated . These errors occur during compilation of the application but in Netbeans these errors are highlighted in design stage itself using the errors indicator . Some of the common examples of syntax errors are missing semicolon , missing parenthesis and using incompatible data types

**Q.5. The code given below will give an error on execution if the value entered in t2 is 0 . Identify the type of the error and modify the code to handle such an error .**

```
int a,b,c;  
a = Integer.parseInt(t1.getText());  
b = Integer.parseInt(t2.getText());  
c = a / b ;
```

**Ans :** The error is logical error .

```
int a,b,c ;  
a = Integer.parseInt(t1.getText());  
b = Integer.parseInt(t2.getText());  
if (b!=0)  
    c = a / b ;  
else{  
    JOptionPane . showMessageDialog (null,"Denominator cann't be zero");  
t2.set Text("");  
t2.requestFocus();  
}
```

**Q .6. What are the characteristics of a good program ?**

Ans : The characteristics of a good program are-

- The program should be efficient in terms of execution speed and effective memory utilization .
- The should be accurate. It should produce correct result.
- The program should user friendly. It means meaningful names should be given to variable, proper messages should be given, use of comments and indentation.
- The program must be reliable that is it should be able to handle the situation when the wrong inputs are given.
- The program should be portable show that it can run no different platforms without doing any changes.

**Q .7. What is the use of comments and indentation ?**

Ans . Comments are non-executable statements and are used for internal documentation purpose . In JAVA comments are given either by // or /\*...\*/ brackets .

Example –

```
/*This method calculates sum of two numbers .*/  
int sum(int x , int y)// x , y are formal parameters  
{ return ( x+y ) ; }
```

Indentation makes a program readable and understandable. When you are writing a program you must remember that

- (i) The opning braces should properly match with a closing braces .
- (ii) Spaces should be inserted between operator and operands in an expression.