decimal both)

JAVA RESERVED WORDS / key words

Java reserved words or the keywords are the words which carry special meaning to the system compiler. These words are basically used for writing a Java statement. Such words cannot be used for naming a variable in the program. Some of the reserved words or keywords are listed as below:

case	switch	else	break	static
do	const	throws	float	char
try	int	double	void	goto
for	while	new	import	boolean
long	if	byte	package	private
catch	short	public	class	default

int - number
void main

Elementary Concept of Objects and Classes

Learning Scope

Introduction, objects; real world objects and software objects, Characteristics and behaviour of real world objects and software objects, Class as a specification of objects, creating objects of a class, message passing, Important terms related to class: class as an object factory, class as a user defined function, declaring states and methods of a class, object as an instance of a class, differences between class and object.

INTRODUCTION

In the previous chapter, you have learned about Object Oriented Programming (OOP) and its principles such as Data abstraction, Encapsulation, Polymorphism and Inheritance. An Object Oriented Programming (OOP) language not only promotes these features but also correlates two more basic units (i.e. Objects and Classes).

In this chapter, we are going to discuss the features or properties of objects and classes applicable to real world as well as to software world.

OBJECTS

An object is a fundamental unit of Object Oriented Programming and represents the real life entities. In Object Oriented Programming (OOP), the attempts are made to break a task into some components, called *objects*. They are the basic elements of the Object Oriented system and are also known as the *Entities*. Moreover, a set of related objects may exchange data and information to interact with each other.

The objects are further categorised in two ways:

(a) Real World Objects

(b) Software Objects

REAL WORLD OBJECTS

Real world objects are those which we experience or use in our day to day lives. A specific item can be defined as a real world object if it possesses following features:

- It is visible to us.
- It has a definite shape and size.

- It can be brought into thoughts and figures.


Tiger


Cycle


Car


Clock

(Real World Objects)

Each real world object contains characteristics and behaviours. The characteristics basically comprises the parts of its body or specifications whereas, behaviour is the purpose of its use or its functions.

For example, A mobile phone (cell phone) can be defined as a real world object that possesses some characteristics and behaviours as explained below.

The characteristics are:

- Model
- Dial pad
- App

The behaviours are:

- It is used to talk.
- It is used to browse the Internet.
- It is also used for net banking.


Mobile phone

Similarly, you can consider the 'Table' as an object.


Table

The characteristics of a table are:

- It has a plain glass top.
- It has four legs.
- It is made up of wood.

The behaviours of a table are:

- It is used to keep copies and books.
- It is used for study purposes.
- It is used to keep glasses and plates.

The objects described above (like the table and mobile) are non-living things. Broadly, speaking, the living things are also considered as real world objects. Under this situation, we may assume each living individual as an objec whose characteristics may be defined as their features and their behaviour ma be defined as the role or function they perform.

For example, a student may be considered as an object.

The characteristics of a student are:

- A student is identified by his/her name.
- Each student studies in a specific class.
- Each student has been allotted a unique roll number.

The behaviour of a student is given by:

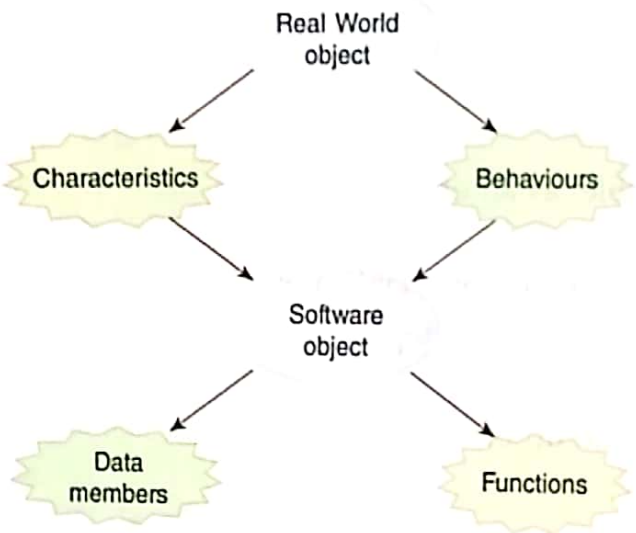
- He/She goes to school.
- He/She appears for the examination.
- He/She takes part in different competitions.

Student

SOFTWARE OBJECTS

A software object may be defined as an object that is created while writing a Java program. The characteristics and behaviours of real world objects are referred to as data members and member functions (methods) of software objects respectively.

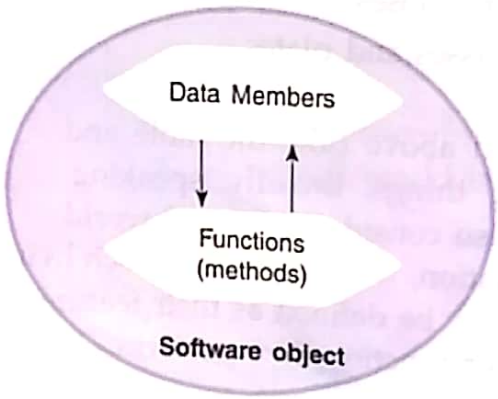
The objects, mobile, table and student, which we have discussed above are real world objects. In fact, a program does not deal with real world objects. A program uses software objects. Let us relate a real world object with a software object.



Real World Objects vs Software Objects

- The states or characteristics of the real world objects are considered to be data members in the software objects.
- The behaviour in the real world objects are referred to as the functions or methods in the software objects.

Hence, a software object is a unique combination of data members and functions (methods), which is shown as:

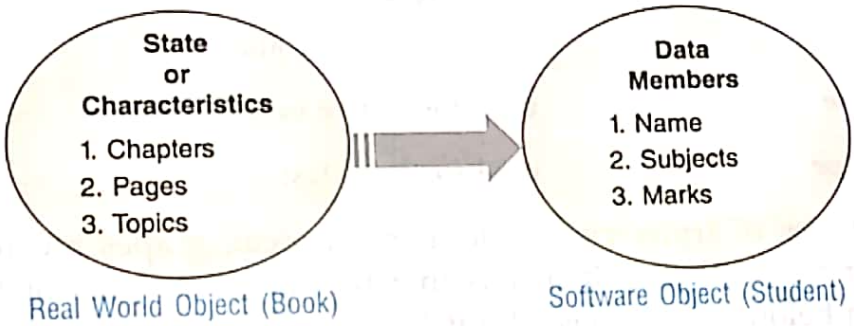


Let us have the comparative study of real world objects and software objects with the help of a table given below:

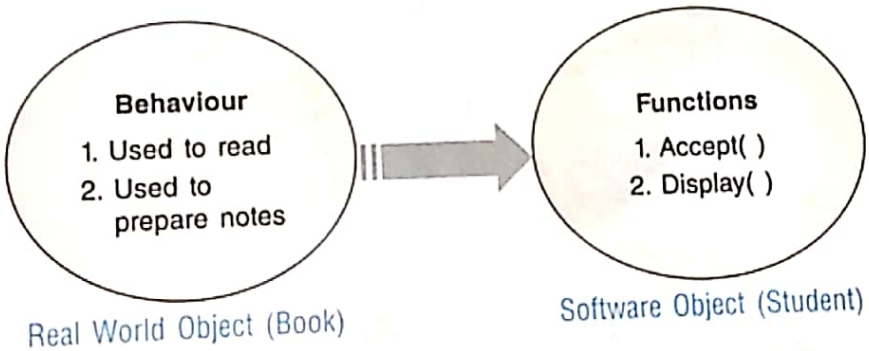
Object	States or Characteristics	Behaviour
Real World (Book)	Chapters Pages Topics	Used to read Used to prepare notes

Object	Data Members	Functions
Software (Student)	Name Subjects Marks	Accept( ) Display( )

A comparative study of characteristics and data members:

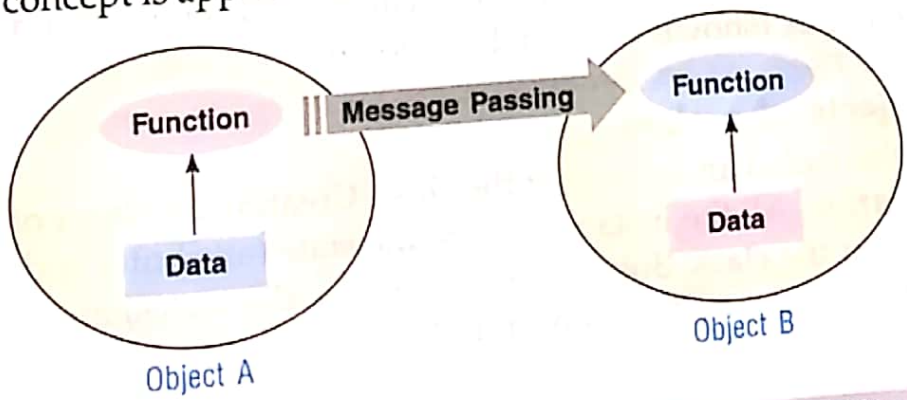


A comparative study of behaviours and functions:



MESSAGE PASSING

It is interesting to know that the objects can interact with each other. An objec can pass information to another object and receive information as well. They interact through behaviour or functions, which are better known as *Messag Passing*. This concept is applicable to real world objects as well as software objects



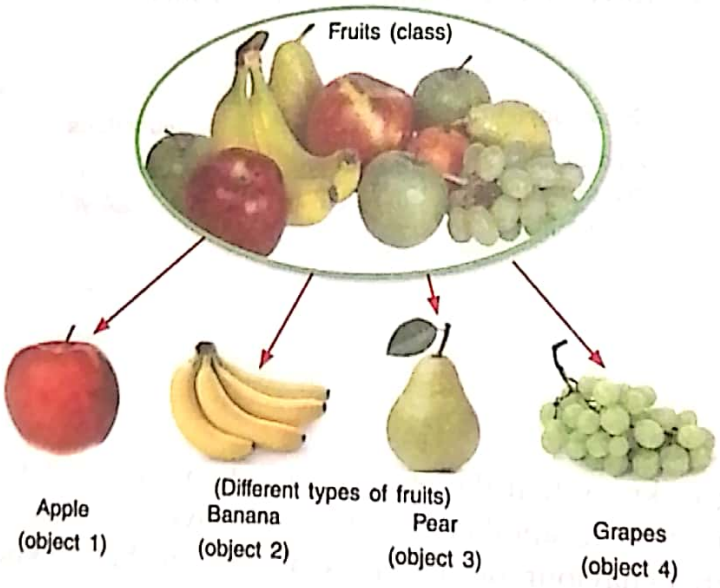
Class

As discussed above, an object is a unique entity possessing some characteristics and behaviour. You might be thinking that from where these characteristics and behavior are embedded in an object'. It is done with the help of a 'Class'. Basically, a 'Class' is a blue print or template for its objects. It can also be defined as a category that describes some characteristics and behaviour. Each object of a class possesses the characteristics and the behaviour (data and functions) described within the class. In this way, a 'Class' and its 'Objects' have a close relationship in an Object Oriented Programming System. They can be considered to be the two sides of the same coin, which cannot be disintegrated. It is insignificant to discuss an object without referring to its 'Class'.

If we define a class 'Fruits', then it may describe the following characteristics and behaviour:

Characteristics	Common behaviour
colour	used for making juice
taste	used for festive occassions
shape	used for breakfast

Different types of fruits viz. apple, banana, pear, grapes are referred to as objects under that category. It means that the different types of fruits are the objects which belong to the class 'Fruits'.



Here, each object of the class 'Fruits' will have different characteristics with common behaviour (shown in the table above).

Creating objects of a class

An object is also called an *instance* of the class. Creating an object of a class is said to be ***instantiation***. All the instances share the state (attribute) and the behaviour described within the class. But, the attributes (i.e. the states) are unique for each object. A single class may have any number of instances.

Syntax of creating an Object of a Class:

<Class Name><Object Name> = new <Class Name>( );

With reference to the class 'Fruits' (described above), different objects can be created as shown below:

Fruits apple = new Fruits();
Fruits banana = new Fruits();

Attributes are the characteristics or data members of an object.

Here, each object of Class 'Fruits' (apple or banana) will possess different characteristics but common behaviour as defined within the Class 'Fruit'. The steps to create an object of a Class are illustrated as:

- **Declaration** : It uses class as a data type along with an object.
- **Instantiation**: Creating an object (apple or banana) is termed as Instantiation.
- **New** : The keyword 'new' is used for allocating space in the dynamic memory for storage of an object.

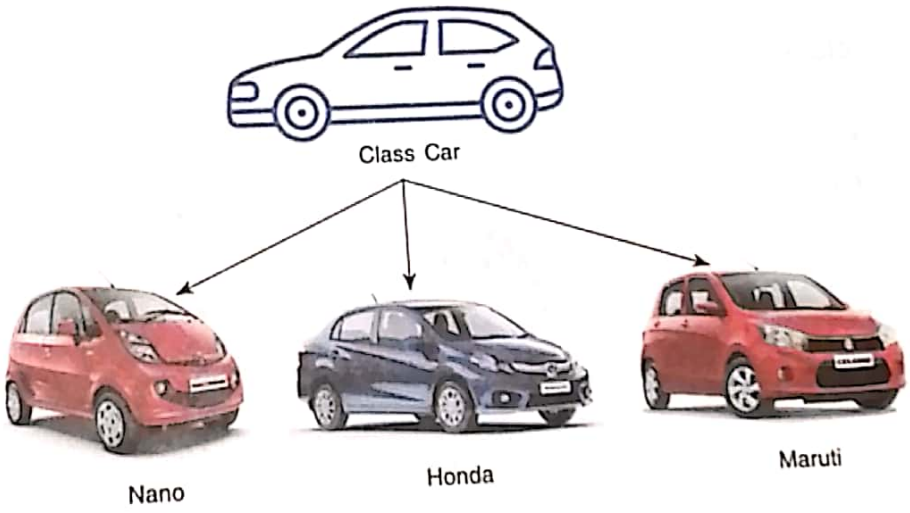
Inter-relation between class and objects

To understand more about the inter-relation of a class with its objects, let us take some examples to create objects by using real world class as well as software class.

Real World Class and Objects

- Let us take an example of a class 'Car'. The characteristics and behaviour can be defined as shown below:

```
class Car
{
Characteristics:
    Colour
    Model
Behaviour:
    Journey
    Demo
}
```



The various objects of the class 'Car' can be created by using the following statements:

Car Honda = new Car( ); : It means Honda will be an object of the class Ca
Car Nano = new Car( ); : It means Nano will be an object of the class Ca
Car Maruti = new Car( ); : It means Maruti will be an object of the class Ca

The new operator

The keyword **'new'** is used to allocate space in the dynamic memory for the storage of data and functions belonging to an object.

As the objects of class Car have been created, they will follow the structure as shown below:

	Honda	Nano	Maruti
Different Characteristics	Colour	Colour	Colour
	Model	Model	Model
Common Behaviour		Journey Demo	

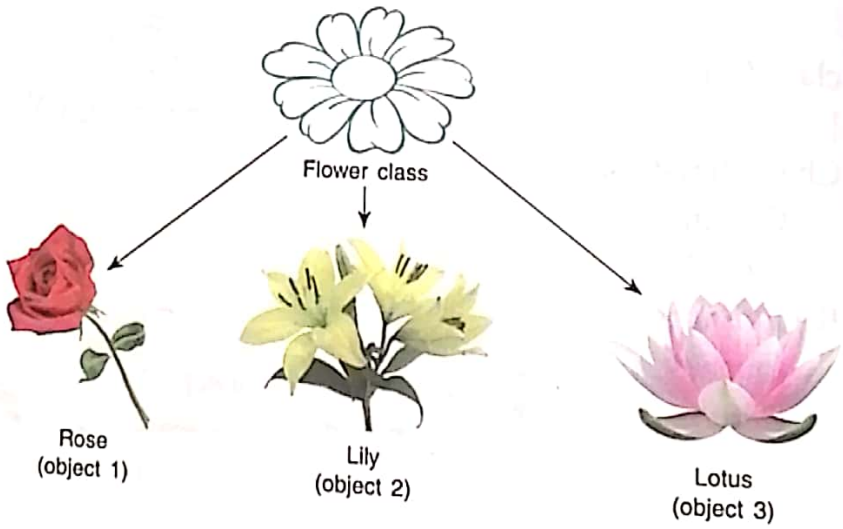
Here, each object of the class 'Car' will possess different characteristics like colour and model but will have common behaviour like Journey and Demo.

If you want to go for a journey on a Honda then the characteristics of Honda will be used. Similarly, it can also be understood for the objects Nano and Maruti.

A **class** is simply a representation of similar types of Objects. It is the blueprint/ plan/ template that describes the details of an Object.

- Let us take another example. Here, we are going to assume 'Flower' as a class. The blue print or template of the class 'Flower' can be designed as shown below:

```
class Flower
{
Characteristics:
Colour
Smell
Behaviours:
Decoration
Worship
}
```



Now, the objects of the class 'Flower' can be created by using the following statements:

```
Flower rose = new Flower();
Flower lily = new Flower();
Flower lotus = new Flower();
```

Here, the objects rose, lily and lotus are referred to as class tags or instances of the class. It means each object of the class 'Flower' will follow the blue print

and will possess the characteristics and behaviour defined within it. Have a look on the table shown below:

Different Characteristics					
Rose		Lily		Lotus	
Colour	Red	Colour	Yellow	Colour	Pink
Smell	Yes	Smell	Yes	Smell	No
Common Behaviour					
Decoration					
Garland					
Worship					

The table shown above illustrates that the objects rose, lily and lotus possess different characteristics but they have common behaviour. The behaviour mainly refers to the functions to be carried out, on different objects. To carry the functions on different objects, the statements can be written as:

rose. Decoration : It means the characteristics (Colour, Smell) of rose will be used for decoration.

lily. Decoration : It means the characteristics (Colour, Smell) of lily will be used for decoration.

lotus. Worship : It means the characteristics (Colour, Smell) of lotus will be used for worship.

Just to Know

A typical Java class creates many objects and interacts by invoking methods. An object consists of:

1. **State:** It is represented by the attributes of an object. It also reflects the properties of an object.
2. **Behaviour:** It is represented by the methods of an object. It also reflects the relation of an object with the other objects.
3. **Identity:** It gives a unique name to an object and enables an object to interact with the other objects.

Software Class and Objects

Now, we will take examples of software class and objects that are used for programming purpose in Java language.

Let, 'Employee' be a class that describes the following data and member methods:

```
class Employee
{
    //Data Members
        Name
        Dept
        Salary
    //Member Methods
        AcceptData( )
        DisplayData( )
}
```

The objects of the class Employee can be created by using the following statements:

```
Employee staff1 = new Employee( );
Employee staff2 = new Employee( );
Employee staff3 = new Employee( );
```

Each object of the class 'Employee' created above will contain different data values for Name, Dept and Salary and common functions (Methods) such as AcceptData( ) and DisplayData( ).

The table shown below illustrates the associativity of data and functions under different objects:

Data Members					
	Staff1		Staff2		Staff3
Name	Amar	Name	Suresh	Name	Peter
Dept	CEDT	Dept	LD Shop	Dept	Automation
Salary	₹45,000	Salary	₹35,500	Salary	₹52,735
Member Methods (Functions)					
AcceptData( )					
DisplayData( )					

Some of the functions that can be carried out on different objects are shown as:

Staff1.AcceptData( ) : It will accept Name, Dept and Salary for object Staff1.
Staff2.AcceptData( ) : It will accept Name, Dept and Salary for object Staff2.
Staff1.DisplayData( ) : It will display Name, Dept and Salary of object Staff1 on the screen and so on...

- Let us design a class 'Product'. The following may be the characteristics and behaviour of class 'Product':

```
class Product
{
    // Data Members
    Quantity
    Rate
    Amount
    // Member Methods
    InputVal( )
    Calculate( )
    Display( )
}
```

The objects of class 'Product' can be created by using the following statements:

```
Product tvset = new Product();
Product cooler = new Product();
```

Once, the objects are created it will possess data members and member methods as shown below:

Data members			
TV set		Cooler	
Quantity	4	Quantity	5
Rate	50,000	Rate	8,000
Amount	2,00,000	Amount	40,000
Member Methods			
InputVal( )			
Calculate( )			
Display( )			

The member methods will be invoked on each object as:

tvset.InputVal( ) : It will accept the Quantity and Rate of the tvset.
cooler.InputVal( ) : It will accept the Quantity and Rate of the cooler.
tvset.Calculate( ) : It will calculate the amount for the tvset.
tvset.Display( ) : It will display the amount for the tvset.
and so on...

Note

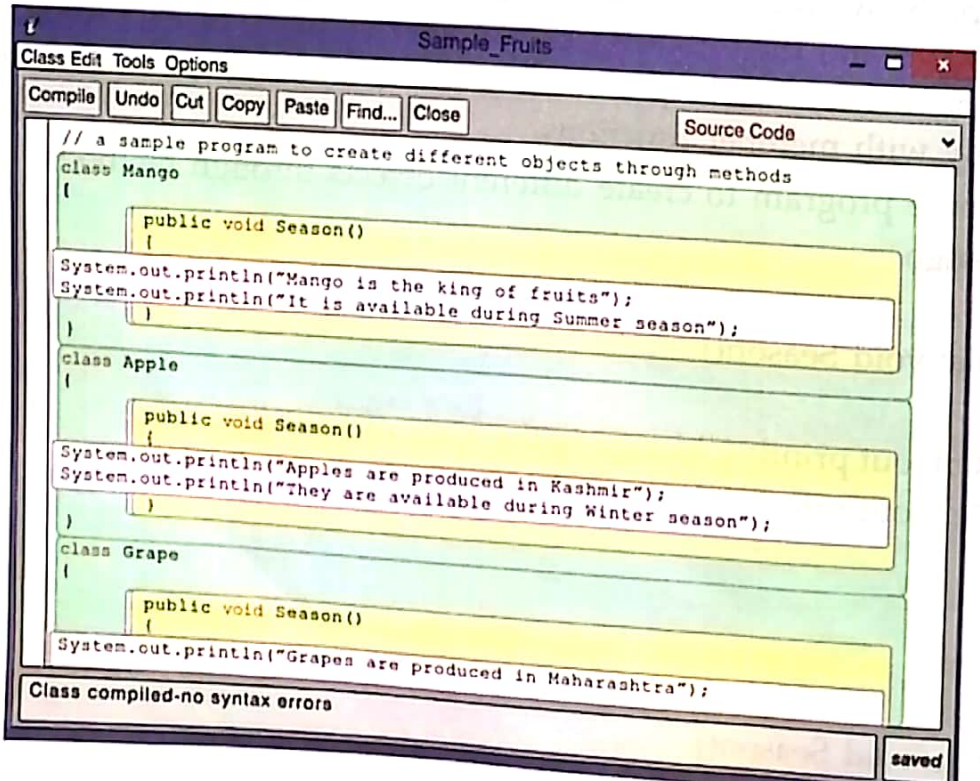
Since, you are not aware of the data types and writing Java statements, in the above examples of software class and objects, neither data types are mentioned nor the methods are defined. The templates and explanations are only for understanding the inter-relationship of class with its objects.

By now, you would have learnt about the class, objects and their inter-relationship. Now, we are going to write an executable program by using different classes and their objects. As you are not aware of data types, variables and constants hence, the classes are available without the data members but are defined only with member functions.

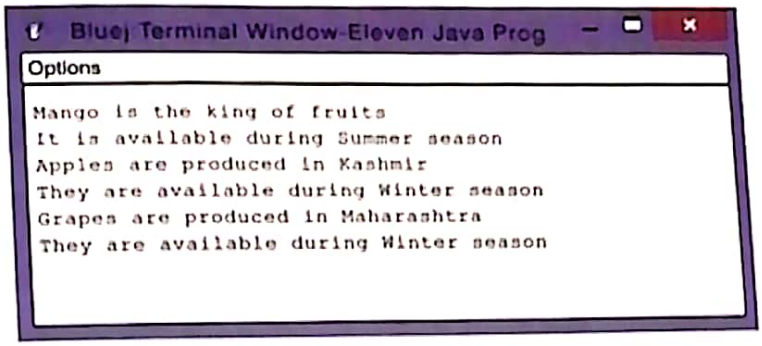
```
// a sample program to create different objects through methods
class Mango
{
    public void Season()
    {
    System.out.println("Mango is the king of fruits");
    System.out.println("It is available during Summer season");
    }
}
class Apple
{
    public void Season()
    {
```

```
        System.out.println("Apples are produced in Kashmir");
        System.out.println("They are available during Winter season");
    }
}
class Grape
{
    public void Season()
    {
        System.out.println("Grapes are produced in Maharashtra");
        System.out.println("They are available during Winter season");
    }
}
public class Sample_Fruits
{
    public static void main(String[] args)
    {
    Mango Fruit1 = new Mango();
    Apple Fruit2 = new Apple();
    Grape Fruit3 = new Grape();
    Fruit1.Season();
    Fruit2.Season();
    Fruit3.Season();
    }
}
```

ompilation of the program:



The Output:



Important terms related to Class

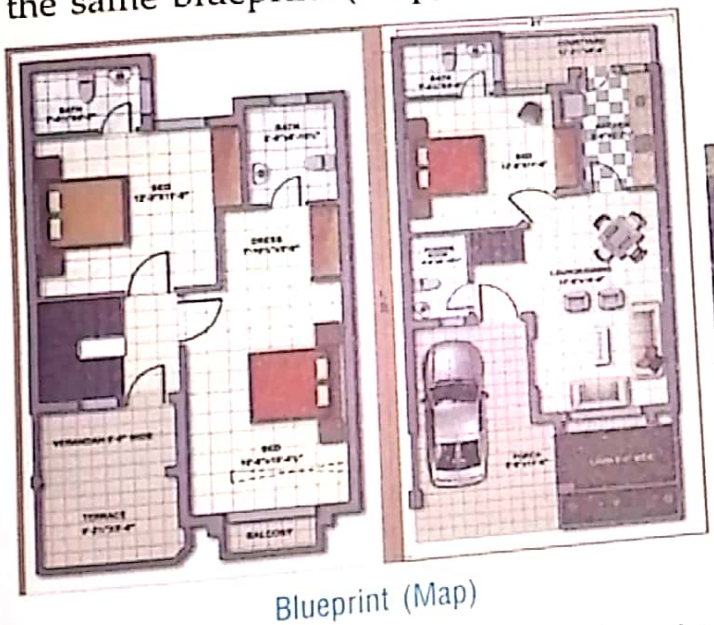
There are some important terms that we come across during the discussion of a Class. They are explained as under:

❖ *Class as an Object Factory*

Generally, the term 'Factory' is used along with a class for creating similar kinds of objects. These objects are created by using all information that share different characteristics and common behaviour, available in the class. When an object is brought into existence, the process is called *Instantiation*.

For example:

While constructing buildings, at first, a blueprint (Map) of a house is prepared to make the real house. It means we can construct many houses (objects) from the same blueprint (Map).



Blueprint (Map)

Houses (Objects)

Thus, we can conclude that class is the prototype of an object. Each obje belonging to a specific class possesses the data and functions defined with the class. It also produces the objects of similar type. Hence, a class is term as 'Object Factory'.

❖ *Class is a user defined data type*

Primitive data types are not sufficient to handle complex operations. In the world, we have much more complicated objects. Object oriented programm

allows us to model real-world objects. User defined classes combine the data and methods as integrated components.

Thus, we can say that Java language uses some pre-defined data types viz. int, float, char, etc. These data types provide some preliminary facilities in Java programming. When they are used to declare any variable, then the variable would possess its built-in characteristics. Hence, these types are called built in data types. Similarly, a user may create a data type and declare certain characteristics and behaviour within it. This can be done by using a class. This is the reason why a class is referred to as *user defined data type*.

❖ *Object is an instance of a class*

In Java language, each object is created from a class and so is called an instance of that class. Creating an instance of a class is sometimes referred to as 'instantiating' the class.

A real-world example of an object would be a 'Doberman', which is an instance of a class called 'Dog'. We have already discussed that the data members of a class are also referred to as instance variables. The instance variables get embedded automatically within an object at the time of its creation. Hence, an object is referred to as an *instance of a class*.

Declaring States (attributes) and Methods of different Objects of a Class

The characteristics (data members) described within a class are also known as States or Attributes for the objects of that class.

Some of the attributes of the objects of a class are explained below:

Class Television	
State (Characteristics)	**Methods (Functions)**
Company, Model, Size, Price (Attributes)	Getdetails( ), Display( )

Class Student	
State (Characteristics)	**Methods (Functions)**
Name, Adm_No., Date_of_birth, Aadhar_No. (Attributes)	Accept( ), Display( )

Differences between Class and Object

Class	Object
1. It is a representation of an abstraction only.	1. It is a real and unique entity having some characteristics and behaviour.
2. It is an object producer and hence called a blue print for a set of objects.	2. It is created with the help of the 'new' operator.
3. It is known as the 'Object Factory'.	3. It is known as an 'Instance of a Class'

REVIEW INSIGHT

(a) Why is an object called an instance of a class? [ICSE-2010]

Ans. Since, an object possesses instant variables and member methods defined within the class. It is called an instance of a class.

(b) What is the difference between an object and a class? [ICSE-2011]

Ans. The differences between an object and class are:

Class	Object
1. It is a representation of an abstraction only.	1. It is a real and unique entity having some characteristics and behaviour.
2. It is an object producer and hence called a blue print for a set of objects.	2. It is created with the help of the 'new' operator.
3. It is known as the 'Object Factory'.	3. It is known as an 'Instance of a Class'.

(c) How are Class and Objects inter-related? [ICSE MODEL]

Ans. A class is used to create various objects which possess different characteristics and common behaviour defined within it. So, we can say that a class is a blue print or a prototype of an object. Thus, each object follows all the features which are defined within the class.

For example, If 'Car' is the class with the characteristics colour, model and version, then Nano, I20 and Swift Dzire can be the objects of the class 'Car'.

(d) Write a Java statement to create an object mp4 of class digital. [ICSE 2013]

Ans. Java statement to create an object mp4 of class digital:

```
digital mp4 = new digital( );
```

(e) Why is a class called an object factory? [ICSE 2009]

Ans. A Class is used to produce or create various objects containing different attributes and common behaviour. Hence, a class is called an object factory.

(f) Why is a class known as a composite data type? [ICSE 2009]

Ans. A class is a tool to create user defined data types. The class name becomes the data type for the instances used in the program. It is such a data type that includes various predefined data types within it. Hence, the class is said to be a composite data type.

(g) What is the significance of using the word 'new' while creating an object? [ICSE MODEL]

Ans. The keyword 'new' is used for dynamic allocation of an object, i.e., it allots space in the dynamic memory for the storage of an object.

(h) What is meant by the term attributes? [ICSE MODEL]

Ans. Attributes are defined as the characteristics or the data members possessed by the objects of a class. It is also referred to as states of an object.

Values and Data Types

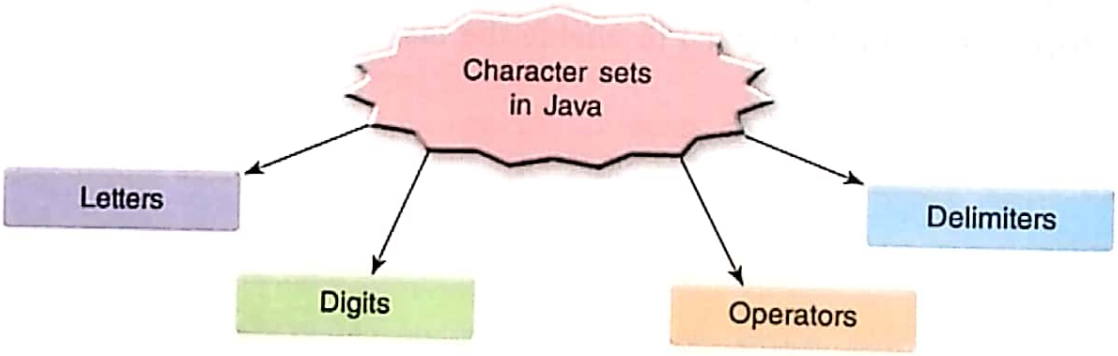
Learning Scope

Introduction, character sets in Java, Unicode, ASCII code, Escape sequences, Tokens, Different type of tokens (Keywords, Literals, Identifiers, Punctuators, Operators), Constants and variables, Initializing a variable, Assignments, Data types in Java: Primitive and Non-Primitive types, Type conversion: Implicit and Explicit type conversions.

INTRODUCTION

A language is a medium of communication. People can share their views with each other, if they know a common language. Hence, language plays an important role in communicating our thoughts and ideas.

Java is a computer language that enables the users to communicate with the computer. It uses character sets similar to alphabets of general languages. The character sets used in Java are as shown below:



CHARACTER SETS

The different character sets are explained as under:

- **Letters** : All the letters of English alphabet (A-Z and a-z) can be used in Java language.
- **Digits** : Digits (0 – 9) can be used in Java language.
- **Operators** : Operators can be classified in three different categories.
  - (*i*) *Arithmetical Operator:* All arithmetical operators like +, -, *, /, % are applicable in Java programming.
  - (*ii*) *Logical operator:* In Java, &&, ||, ! are used as logical operators.
  - (*iii*) *Relational operators:* The Java language uses relational operators as <, <=, >, >=, == and !=.

- **Delimiters :** Delimiters are the special characters. In Java language, delimiters are used as -, ; :, ?, ., (,), {, }, [,], etc.

UNICODE

When a character is entered from the keyboard, a specific code is assigned by the system for its storage in the memory. Earlier the ISO and IEC coding schemes which were in use represented only a limited number of characters. Nowadays, the latest coding system of the characters followed world wide, is known as the *Unicode*.

> Hence, Unicode is a standard encoding system created by Unicode consortium that is used to encode a character in any computer language.

The Unicode is a wide representation of characters in the numeric form. The code contains hexadecimal digits ranging from 0x0000 up to 0xFFFF (i.e. 16-bits code). It can address 1, 60, 755 characters in computer from 139 modern and historic scripts. It means a code of each character is available under the Unicode character set.

Advantages of using Unicode

The advantages of the character coding scheme by using the Unicode are shown as:

- It is the universal coding scheme followed throughout the world.
- It is a more efficient coding system than the ISO or IEC.
- It supports uniform coding width for all the characters (16-bits).
- A particular code of a character is always unique, i.e., there will never be one code for more than one character.

The table shown below illustrates some characters represented by their Unicodes:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0020		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0030	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0040	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0050	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0060	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0070	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

ASCII CHARACTERS AND CODES

ASCII stands for *American Standard Code for Information Interchange*. The characters used in the computer are said to be ASCII characters. Each character is assigned a specific numeric value, called the ASCII code. The ASCII codes are the decimal numbers represented in 7 binary digits. The range of code is not so large as compared to Unicode. ASCII codes are available only for limited number of characters ranging from 0 to 127. Basically, the ASCII codes are used while programming, when it deals with characters.

Some ASCII codes of well-known characters are as listed below:

ASCII Codes	ASCII Characters	ASCII Codes	ASCII Characters
48	0	86	V
49	1	87	W
50	2	88	X
51	3	89	Y
52	4	90	Z
		******	******
53	5	97	a
54	6	98	b
55	7	99	c
56	8	100	d
57	9	101	e
******	*******	102	f
65	A	103	g
66	B	104	h
67	C	105	i
68	D	106	j
69	E	107	k
70	F	108	l
71	G	109	m
72	H	110	n
73	I	111	o
74	J	112	p
75	K	113	q
76	L	114	r
77	M	115	s
78	N	116	t
79	O	117	u
80	P	118	v
81	Q	119	w
82	R	120	x
83	S	121	y
84	T	122	z
85	U		

Note

(i) 58-64 are different symbols such as <, >, :, etc.
(ii) 91-96 represents some special symbols viz. [,], \, ^, etc.
(iii) ASCII code of white space (blank) is 32.

Difference between Unicode and ASCII code

Unicode	ASCII Code
1. Unicode is a generalised form of coding scheme for numerous characters of different scripts.	1. ASCII code is a specific coding scheme used for limited characters.
2. Unicode represents a higher range of codes.	2. ASCII code represents a limited range of codes.

Escape Sequences

There are some non-graphic characters, which are used as commands to direct th cursor while printing. These characters are frequently used in Java programmin and are called *Escape Sequences*. An escape sequence character begins with backslash (\) and it is followed by one or more characters. This is the reas why escape sequences are also called *back slash characters*. A table is given f your reference.

Escape Sequences	Non-Graphic Character
\t	Horizontal tab
\\	Backslash
\'	Single quote
\"	Double quote
\b	Backspace
\f	Form feed
\0	Null
\r	Carriage return
\n	New line feed

Using Escape Sequences

Basically, escape sequences are available in Java programming to contr cursor's movement on the screen. It allows the user to customise the scree get a formatted output. Escape sequences are used with print statement en within double quotes.

Some escape sequences which are commonly used in Java programmi explained below:

(*i*) "\n" (Backslash n): This character is used for a new line feed. A as a "\n" is encountered, the cursor skips the current line and m the next line on the screen for printing the remaining part of the

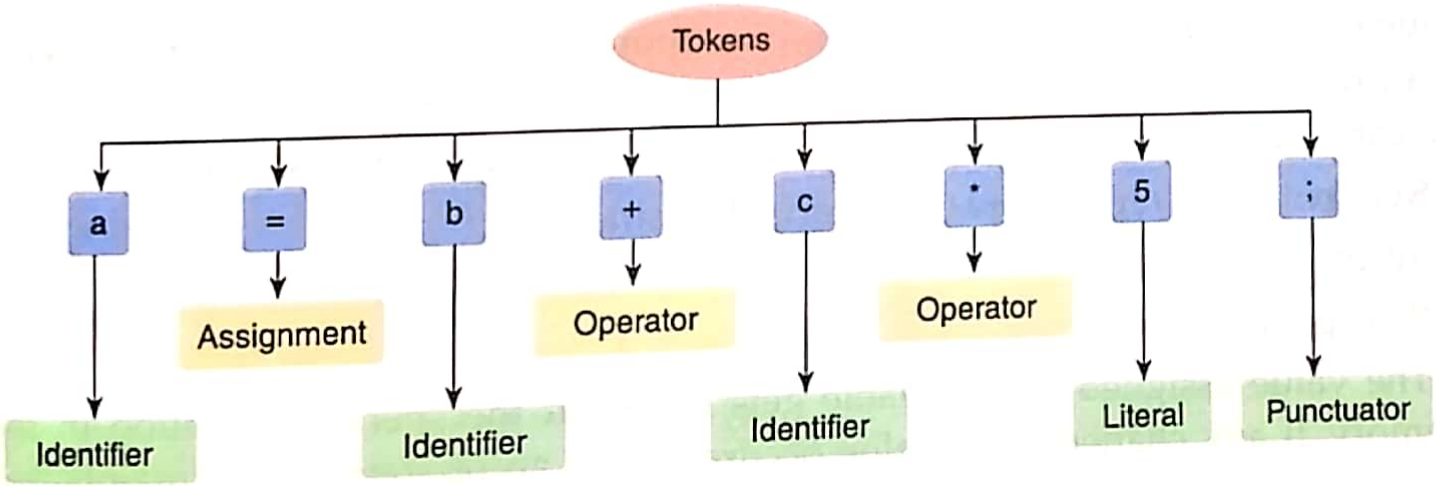
Token

You would have studied in biology about the 'cell' which is a fundamental and a functional unit of the human body. Like a cell in the human body, a token is also a functional and a fundamental unit of a computer program. You know that a computer program is a set of statements. A statement is composed of various components and each individual component of a programming statement is referred to as a *Token*.

The various types of tokens available in Java are:

- Literals
- Identifiers
- Assignments
- Punctuators
- Separators
- Operators
- Keywords

A token can be defined as each individual component of a Java statement such that it carries some meaning and takes part in effective execution of the program.



Literals (Constants)

Literals are the constants in a Java program. When you write a program in Java, you may come across some quantities, which remain fixed (i.e. do not change) throughout the execution of the program. Such quantities are termed as *Literals* or *Constants*.

Java literals are classified as under:

- **Integer Literals:** The numbers which are represented without decimal points are called Integer Literals. They are the whole numbers having positive or negative values. For example, 14, 345, 8, 6392, –18, –391, etc.
- **Real Literals:** Real literals are also called floating-point constants. They represent numbers with decimal points.
  For example, 24.6, 0.0072, –3.652, 1.0E-03, etc.
- **Character Literals:** The constants, which are alphanumeric in nature, are called character literals. All letters (upper case or lower case), digits, special symbols can be termed as character literals.
  For example, 'A', 'd', '3', '*', etc.
  A character literal represents a single character enclosed within single quotes.
- **String Literals:** A string is a set of alphanumeric characters. A group of characters enclosed within a pair of opening and closing double quotes is known as a string literal.
  For example, "COMPUTER", "Year 2016", "10% per annum", etc.
- **Boolean Literals:** Boolean constants are special literals. They represent true or false and can be used in a Java program to check whether a given logical condition is satisfied or not. You must note that boolean constants (i.e. true or false) are never enclosed within quotes. This characteristic makes boolean constants different from string constants.
- **Null Literal:** This is also a special purpose literal. It is represented as **null** (each letter in lower case) and is used to initialize an object or reference variable.

Identifiers (Variables)

The identifier is a term used to represent program elements such as function name or class name. The variables used in Java programming are also called identifiers. A variable is a named memory location, which contains a value. The value of a variable can change depending upon the circumstances and problems in a program. A variable can possess any combination of letters without spaces. We can declare more than one variable of the same type in a statement.

Syntax: <data type> <space> <variable name>

```
int m;
float p,q,r;
```

The value assigned to a variable gets stored at the specified location in the memory. If any change in its value occurs due to an operation, it is maintained

in the same location by replacing the existing value. Thus, a variable can change its value as shown below:

Before Execution	After Execution
int m = 5; → m → 	int m = m*m; → 
Data value 5 is stored in location named 'm'	The value of the variable 'm' changes in the memory after the execution

Rules for naming a variable

- A variable may have any number of characters.
- It may contain alphabets, digits, dollar sign and underscore.
- The underscore can be used in between the characters to separate the words of a variable name.
- The variable names should be meaningful, which easily depicts its purpose.

Assignments

Assigning means to store constants in variables using a token '=' symbol. Here, the symbol '=' acts as an *assignment operator*. The data type of the variable depends upon the type of assigned constant. You must ensure that the constant you are going to store has the same data type as the variable declared.

Syntax: Data type <variable> = <constant>;

For example, int m = 15; : 15 is stored in the variable m which is integer type

float n = 45.24; : 45.24 is stored in the variable n which is float type

char chr = 'k'; : The data type char is used to store a character enclosed within single quotes (' ') in a variable chr.

String str = "Computer Applications";

: The data type String is used to store a word/a sentence/a paragraph in the variable name (str), enclosed within double quotes (" ").

The following table illustrates the different data types along with the assignment of relevant constants:

Declaration of variable	Assigning constant
int a;	a = 5;
long b;	b = 2345763;
float f;	f = 3.45;
double d;	d = 0.000000045;
char ch;	ch = 'k';
String str;	str= "COMPUTER";
boolean p	p = false;

Initializing a variable

When you declare a variable, it may contain garbage values (absurd value) and can create a problem during the execution of your program. You may not obtain an appropriate result even though your program logic is correct. Under such circumstances, you need to initialize a variable to get the desired result. A variable is initialized by assigning a specific value to it.

Initialization of a variable takes place in the following ways:

- Static Initialization
- Dynamic Initialization

Static Initialization

This process uses direct assignment of a constant to a defined variable. The variable is initialized at the time of its declaration (i.e., before its actual use in the program logic). The table for the same is as shown below:

Data Type	Declaration	Static Initialization
Integer	int a;	a=0;
Float	float f;	f=0.0;
Double	double d;	d=0.0;
Character	char c;	c= '\u0000';
String	String s;	s= "";
Boolean	boolean p;	p=false;

Dynamic Initialisation

When a variable gets initialised at run time, i.e., during the execution of program logic, it is termed as *Dynamic Initialisation*. Under this situation, a variable is assigned with the outcome of any arithmetical operation or function. The table is as shown below:

Data Type	Declaration	Dynamic Initialisation
integer	int a,b,c;	c=a+b;
float	float p=2.2,k=4.22,f;	f=p+k;
double	int a=49; double d;	d=Math.sqrt(a);
String	String st1, st2, st3;	st3=st1+st2;

Punctuators

Punctuators are the punctuation signs used as special characters in Java. Some of the punctuators are:

(i) ? (question mark) (ii) . (dot) (iii) ; (semi colon)

Question mark (?) represents the action to be taken when the given condition is true while using the ternary operator.

For example, max = (a > b)? a : b;

Dot (.) is used to represent the scope of a function i.e. a function belonging to an object or a class.

For example, (i) System.out.println();
(ii) java.io.*, etc.

Semi colon (;) is used in a Java program as a statement terminator. It indicates the end of a statement. Any line continued after the semi colon is treated as the next statement.

For example, int a =5; System.out.println(a); are treated as two separate statements in Java.

Separators

They are the special characters in Java, which are used to separate the variables or the characters.

For example, Comma(,), Brackets ( ), Curly brackets { }, Square brackets [ ], etc.

Comma (,) in a Java program, is used to separate multiple variables under the same declaration.

For example, int a,b,c;

Brackets ( ) are used to enclose any arithmetical or relational expressions.

Curly brackets { } are used to enclose a group of statements under a compound statement.

Square brackets [ ] are used to enclose subscript or the cell number of a dimensional array.

Operators

Operators are basically the symbols or tokens that perform arithmetical or logical operations. Basically, there are three types of operators used in Java:

(a) Arithmetical Operators: +, -, /, *, etc.
(b) Relational Operators: <,>,= =, !=, <=, etc.
(c) Logical Operators: &&, | | , !, etc.

Keywords

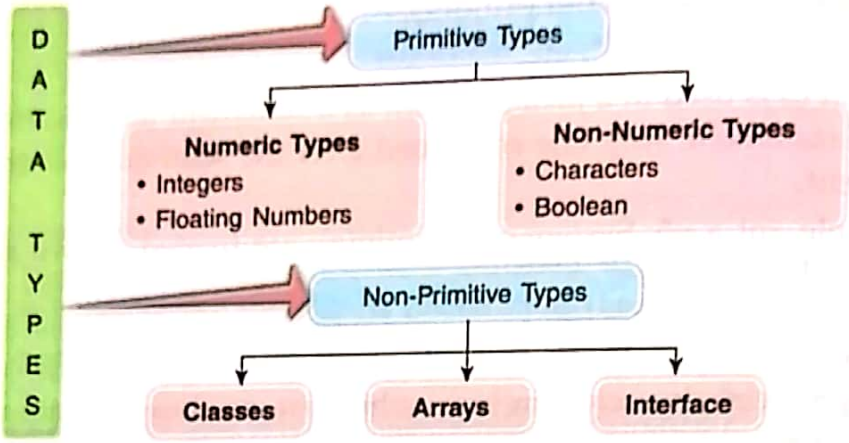
Keywords are the reserved words which are preserved by the system and carry special meaning for the system compiler. During the course of programming you need to use keywords to meet certain requirements.

For example, class, public, throws, for, sqrt, System, etc.

DATA TYPES IN JAVA

The compiler contains a phase called the storage assignment phase. This phase allocates memory for different variables used in your program. It also creates the structure in the location to store the data efficiently. Hence, the compiler must know the type of data you are likely to supply for storage to ensure optimum

utilisation of memory space. This is the reason why data types are required i
Java programming. In Java programming, we need to deal with various type
of data. Hence, it becomes necessary for a programmer to select an appropria
data type according to the data taken in a program. The data and its types a
given below:



Primitive Types

The data types which are independent of any other type, are known as Primitiv
data types. These types are also called Basic Data Types.

For example, byte, int, long, float, double, etc.

Primitive data types are pre-defined or built-in data types because the syste
developers of Java have defined them. You can declare a variable of type in
which will follow the characteristics mentioned in this type.

For example, int x;

It means variable x follows the characteristics of the int type. Hence, variab
x will contain only integer value.

Let us discuss primitive data types in detail:

Integer Type

A variable declared as an integer type contains a whole number. The numbe
may be a positive or a negative number, but without decimal point. There ar
four types of declarations under this heading:

- **byte** : Used for bit-wise operations
- **short** : Used for a small range of integers
- **int** : Used for integers which are more than short integers
- **long** : Used for large integers

A programmer has to select an appropriate data type according to the nee
of the program, as shown below:

Data	Data Type	Bit Size	Format
Byte	byte	8 bits (1 byte)	byte a; a=5;
Short	short	16 bits (2 bytes)	short b; b=12;
Integer	int	32 bits (4 bytes)	int c; c=214;
Long Integer	long	64 bits (8 bytes)	long d; d=45687;

Floating type

When you need to store a fractional number (a number with decimal points), then we declare a variable of the floating type. You can define these numbers in two different ways to represent the data values.

- **float:** It represents a fractional number with a small range of values.
- **double:** It represents a fractional number with a wide range of values.

A programmer decides the data types according to the need. Refer to the table shown below:

Data	Data types	Bit size	Format
Small range of decimal values	float	32 bits	float m; m=34.45;
Wide range of decimal values	double	64 bits	double n; n = 12.1269387;

Characters

A character type variable contains a single character. There are 256 ASC characters out of which only 128 characters are in use. Each ASCII charact is assigned a specific numeric value called ASCII code. The ASCII codes of t characters range from 0 to 127.

Some well known characters and their ASCII codes are as follows:

A – Z : 65 - 90
a - z : 97 - 122 (Respectively)
0 – 9 : 48 - 57

Other codes are used for special characters.

Java language considers a single character and a set of characters (i.e. Str differently. Each character is assigned an ASCII code, which is taken consideration during Java programming. In Java, the declaration of a chara is explained as:

Syntax: <Data type><variable> = <'character literal'>;

```
char p = 'm';
↓    ↓     └→ a character which is assigned to variable p
data type  variable
```

Note: It must be noted that a character is always enclosed within single quot

Similarly, a String (set of characters) is declared as:

Syntax: <Data type><variable> = <"String constant">;

```
String p = "Computer Applications with BlueJ";
↓      ↓     └→ a String which is assigned to variable p
data type  variable
```

Note: It must be noted that a String is always enclosed within double quotes

The character types in Java are as follows:

Non-numeric	Character type	Bit size	Format
Single character (a letter or a special character).	char	16 bits (2 bytes)	char p; p='A'; char x; x='*';
More than a character/a word/a sentence.	String	More than 16 bits	String str; str="School";

Arithmetical Expression and Statement

A set of variables, constants and arithmetical operators used together to yield meaningful result is known as an *Arithmetical Expression*. When an arithmetic expression is assigned to a variable then it is called an *Arithmetical Statement*.

For example,

Arithmetical Expression

d = b * b – 4 * a * c;

Arithmetical Statement

Type of Arithmetical Expressions:

Based on the data types, the arithmetical expression is of the following two type

- Pure Expression
- Impure Expression

Pure Expression

An arithmetical expression that uses all its components of same data types known as the *pure expression*.

For example,

```
int a,b;
int c = a + b * 4;
```

In the expression shown above, all its components like a, b and 4 are intege type data. Hence, it is a Pure Expression.

Impure Expression:

An arithmetical expression in which one or more components are of differen data types, is referred to as an *Impure Arithmetical Expression* or *Mixed mod Expression*.

For example,

```
int a; float f; double d;
double s = a * f / d;
```

Here, a, f and d are of different data types. Hence, it is an impure expression

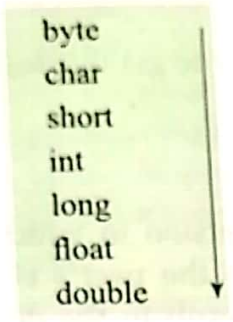
Type Conversion

In a mixed expression, the result can be obtained in any one form of its data types. Hence, it is needed to convert the various data types into a single type. Such conversion is termed as *Type Conversion*. In Java, type conversion takes place in the following two ways:

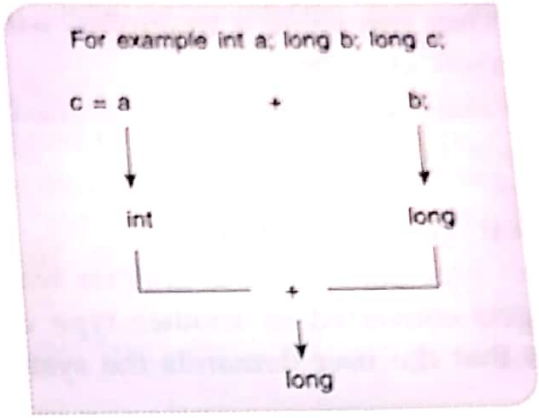
nplicit type conversion

ı a mixed expression, the data type of the result gets automatically converted ıto the highest data type available in the expression without any intervention f the user. This system of type conversion is known as *Implicit type conversion* r *Coercion*.

lierarchy of Data types



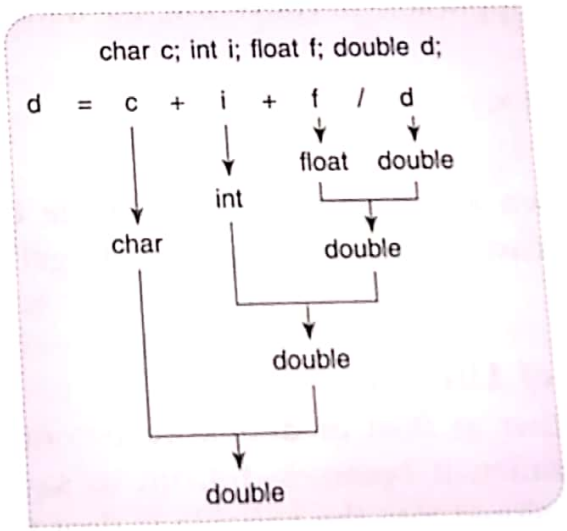
(Hierarchy of the data types)



The hierarchy shown above, indicates the increasing order of the data types. If two data of different data types are operated upon then the result automatically gets converted to their highest data type.

You must have noticed that two values int and long type in the expression shown above will result in the higher 'long type' value. Hence, you must declare variable c as long type.

Consider the following illustration:



The result obtained in the illustration shown above is of the type double. He the resulting variable d must be a double data type.

It is further to be noted that Java Language is very powerful on data operation. So, the user must be very careful of using suitable data types v writing any Java expression.

Suppose, the user wants to calculate area of a trapezium. The data types an the formula to calculate area are shown below:

```
float a, b, h, area;
area = 1/2 * (a + b)*h;
```

At the time of execution, you will get confused, as the area will always resu in zero for whatever may be the values of the variables a, b and h. This because the result of the division (1/2) is implicitly an integer type (i.e., 0 n 0.5). When this value is multiplied with the other data values, it will yield th final result as 0 (zero).

The above expression can be rectified as shown below to get the desired resu

```
area = 1.0/2.0 * (a + b) * h;
```

Explicit Type Conversion

Explicit type conversion is another way of type conversion in which the da ype gets converted to another type depending upon the user's choice. Th neans that the user demands the system to get the result in the desired da ype.

When the data type gets converted to another data type after the user ntervention, the type conversion is known as *explicit type conversion*.

For example, int a,b;

float x = (float) (a+b);

In the example given above, the outcome of the expression (a+b) has to be i nplicitly. But, (float) provided by the User before the expression (a+b) will caus ie result to be forcibly converted to float type. Hence, this type of conversio also called *type casting*.

Type casting is also applicable in converting the data type from a higher typ a lower type.

For example, double x,y;

int c = (int) (x+y);

The expression shown above will result in double type implicitly, but du implication of type casting the result is forced to get converted into a lowe pe int.

plicit Conversion of Literals

variable declared either as float or double type contains a fractional value the time of initialization it becomes difficult to say whether a real litera ctional value) to be stored into the variable is float type or double type. or example,

```
oat f=0.214;
ouble d= 0.214;
```

he example shown above, initializes the same value 0.214 to the floating as l as double type variables. How will you decide whether the value 0.214 is oating constant or a double type constant?

o sort out the problem, explicit conversion of literals is allowed in Java gramming. The literal to be initialized into the floating type variable and ble type variable, must be suffixed with 'F' and 'D' respectively.

or example,

loat f=0.214F;

louble d=0.214D;

Jow, it is clear that the value 0.214F is a floating literal assigned to the variable id 0.214D is a double type literal assigned to variable d.

A similar problem may arise during the initialization of integer and long type iables. A whole number using a suffix 'L' is referred to as a long type literal ereas a whole number without any suffix is an integer type literal.

For example,

nt a=12;

ong b = 12L;

uick Recap

Justify with reason whether the following assignments are true or false:

(i) int n =15.4;

Ans. False, the variable n must be assigned an integer value.

(ii) float f =12.02;

Ans. True, the variable f is assigned a fractional value.

(iii) char ch = 'A';

Ans. True, the variable ch contains a letter enclosed within single quotes.

(iv) String str = "Mouse";

Ans. True, the String constant "Mouse", assigned to variable str, is enclosed within double quotes.

(v) String p = true;

Ans. False, the variable p should not be assigned a boolean constant.

(vi) boolean m = "true";

Ans. False, the variable m should not be assigned a String literal. i.e.; "true" treated as a String constant.

(vii) char p= 'Application';

Ans. False, the variable p must contain a single character within single quot

Operators in Java

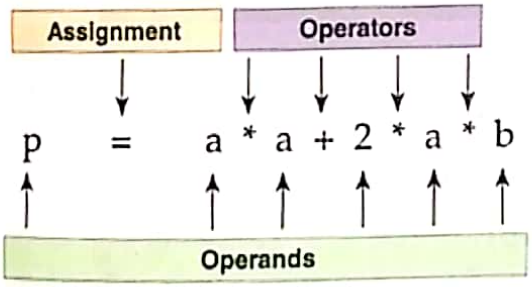
Learning Scope

Introduction, Arithmetical expression and statement, Types of operators: Arithmetical (Forms of operators: Unary (Increment and Decrement, Postfix and Prefix operators), Binary and Ternary), Logical and Relational, New operator, Invoking members of class using dot operator, Output statement: System.out.print() and System.out.println(), Java programs on assignment statement Hierarchy of operators.

INTRODUCTION

In computer programming, you often need to perform some arithmetical or logical operations. In such circumstances, you need operators to perform these tasks. Thus, an operator is basically a symbol or token, which performs arithmetical or logical operations and gives meaningful result. The values, which are involved in the operation, are termed as *operands*.

Let us consider an expression as:



An operator is basically a symbol or token, which performs arithmetical or logical operations to give meaningful result.

ARITHMETICAL EXPRESSION

An arithmetical expression contains variables, constants and arithmetical operators put together, to produce a meaningful result.

For example,

(*i*) x + y
(*ii*) m – 15
(*iii*) a*a+ 2*a*b + b*b
(*iv*) b*b – 4*a*c

ARITHMETICAL STATEMENT

When an arithmetical expression is assigned to a variable then it is known as an *Arithmetical Statement*.

For example,

(*i*) m= x + y
(*ii*) p= m – 15
(*iii*) n= a* a+ 2*a*b + b*b
(*iv*) d=b*b – 4*a*c

TYPES OF OPERATORS

Basically, there are three types of operators to perform any operation in Java programming. They are:

1. Arithmetical Operator
2. Relational Operator
3. Logical Operator

1. Arithmetical Operators

The operators which are used to perform arithmetical calculations in a program are known as *arithmetical operators*. Some basic calculations like addition, subtraction, multiplication, division and modulus (to get remainder when a number is divided by the other number) are often needed during programming. You can use the arithmetic operators like +, -, *, / and % respectively to carry out these calculations.

When you write a program in Java, it is necessary to represent the arithmetical expressions into a Java expression using different arithmetical operators.

Here, few examples are illustrated as how mathematical expressions are written in Java.

Arithmetical Expressions	Java Expressions
abc	a*b*c
ab – bc + ca	a*b – b*c + c*a
$a^2 + b^2 - c^2$	a*a + b*b – c*c
2(l+b)	2*(l+b)
$\frac{prt}{100}$	p*r*t/100
$\frac{1}{3}ab + \frac{1}{2}cd$	1.0/3.0*a*b + 1.0/2.0*c*d

A sample program to find the value of the expressions:

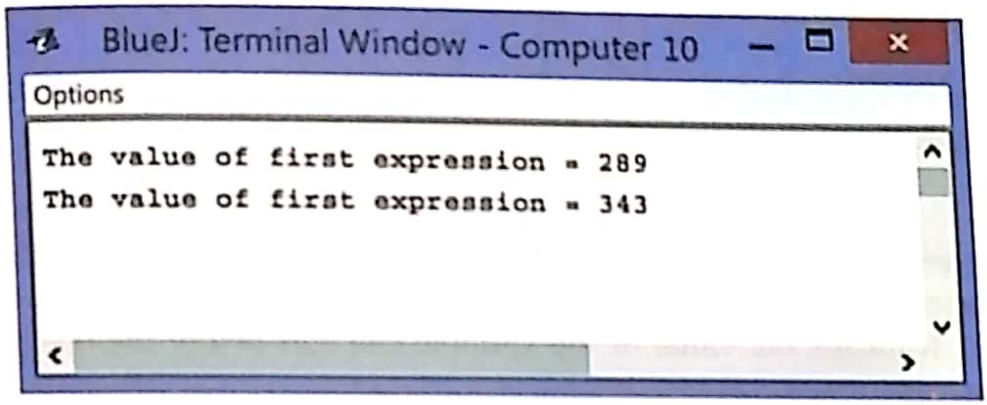
i. $a^2 + 2ab + b^2$

ii. $a^3 - 3a^2b + 3ab^2 - b^3$

```
// A program to display the value of expressions
class Expressions
{
    public static void main(String args[])
```

```
    {
        int a=12,b=5,ans1,ans2;
        ans1=a*a+2*a*b+b*b;
        ans2=a*a*a-3*a*a*b+3*a*b*b-b*b*b;
        System.out.println("The value of first expression = "+ans1);
        System.out.println("The value of second expression = "+ans2);
    }
}
```

Output:



Further, an arithmetical operator can be categorized in the following ways to perform some Mathematical tasks. They are:

a. Unary Operator
b. Binary Operator
c. Ternary Operator

a. Unary Operator

An arithmetical operator, which is used to operate on a single operand, is known as the **Unary Operator.**

For example, +, –, ++, – –, etc. It is further categorised in two ways.

- *Unary (+) Operator*

This operator is used before the operand, simply as a pointer to a variable, which results in the same value of the variable.

For example, If a = 8, then + a will result in 8
If a = –10, then + a will result in –10

- *Unary (–) Operator*

This operator is used in the same way as Unary plus (+). It is also used before the operand. Unary minus (–) operator reverses the sign of an operand.

For example, If a = 4, then – a will result in –4
If a = 0, then – a will result in 0 (It is signless value)
If a = –3.6, then – a will result in 3.6

Unary increment and decrement operators

Unary increment operator (++) increases the value of an operand by one. Una
decrement operator (--) decreases the value of an operand by one.

Examples:

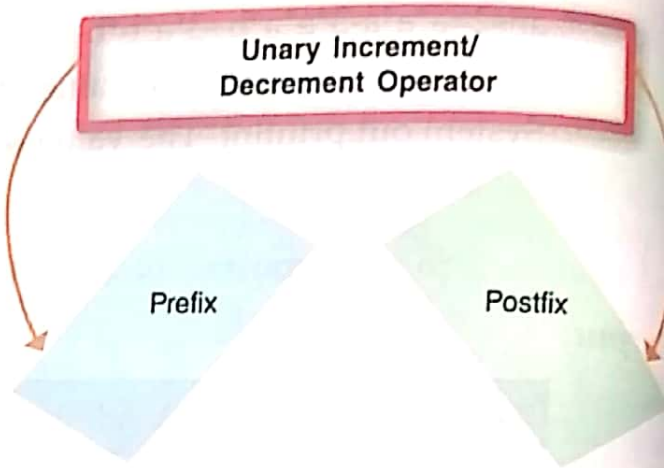
i. x = x + 1

By using increment operator it can be written as x++ or ++x

ii. p = p – 1

By using decrement operator it can be written as p-- or --p

This increment of decrement is used in two different forms.

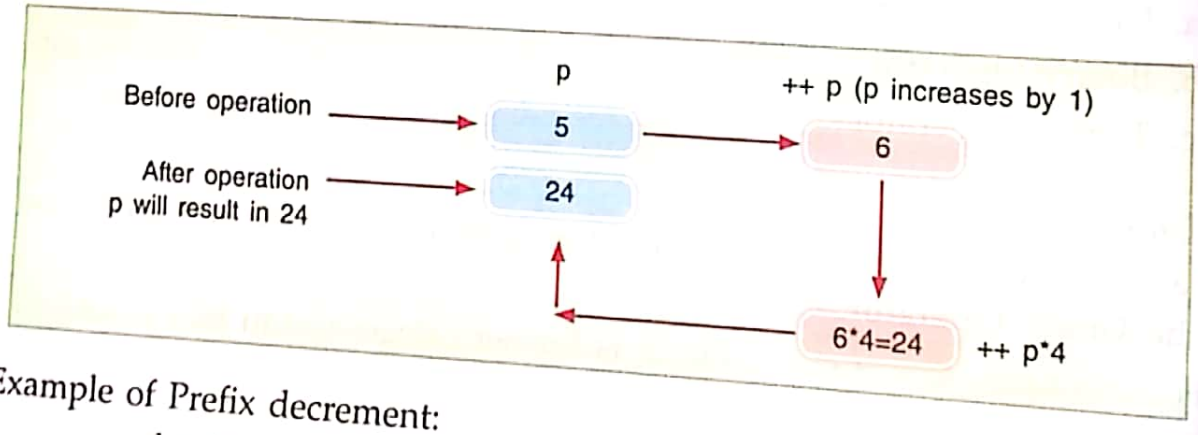


• *Prefix*

When increment or decrement operator is used before the operand, it is know as *prefix operator*. This operator works on the principle of 'CHANGE BEFOR ACTION'. It means the value of the variable changes before the operation wil take place.

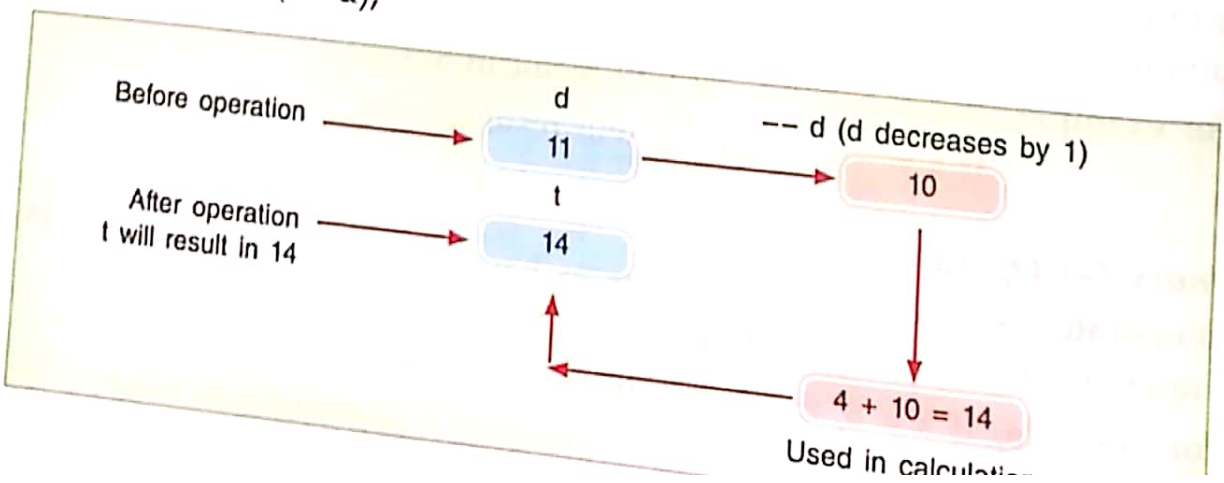
Example of Prefix increment:

```
p = 5;
p = ++p * 4;
```



Example of Prefix decrement:

```
d = 11;
t = 4+(-- d);
```

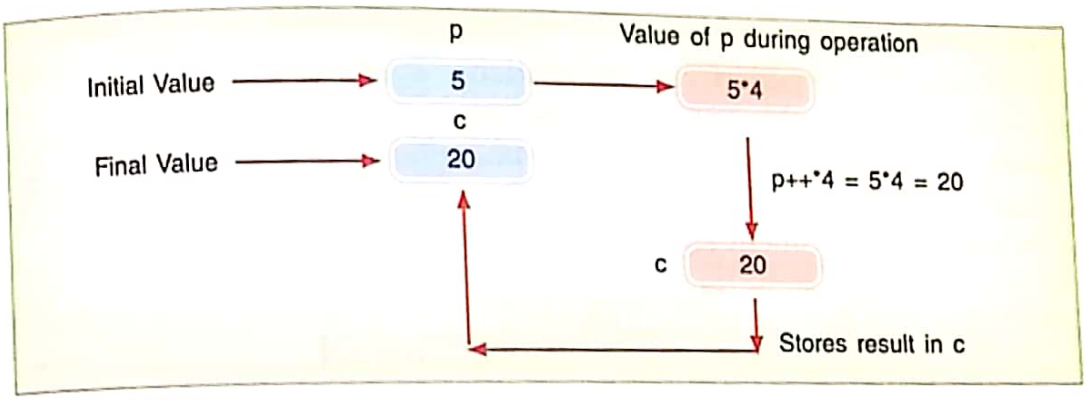


• *Postfix*

This unary operator is used after an operand (whose value is to be increased or decreased by 1). This works on the principle of 'CHANGE AFTER THE ACTION'. This means that the operand will be affected after performing the operation.

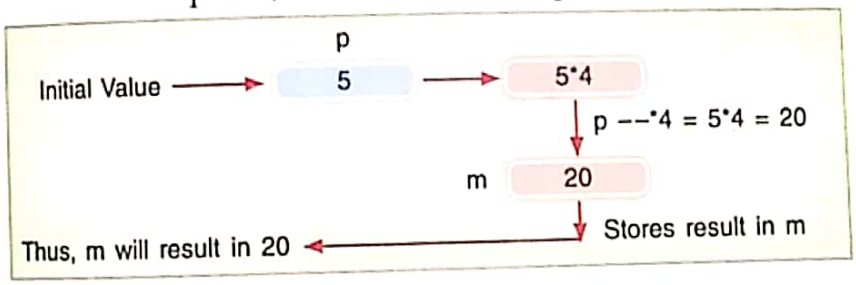
Example of postfix increment:

```
p = 5;
c = p++ * 4;
```



Example of postfix decrement:

```
p = 5;          m = p -- * 4;
```



Note

Both Prefix and Postfix increment operators increase the value of an operand by 1, **before the action and after the action** *respectively. Similarly, Prefix and Postfix decrement operators decrease the value of an operand by 1.*

Counter and Accumulator

Counter

The counters are numeric variables, which are used for keeping a record of the number of times a process is repeated. A counter is initialized with a numeric value and it increases as the process starts repeating.

For example, `c = 1;`

`c = c + 1;` or `c++;`

Here, the value of 'c' will increase by '1', the moment statement `c = c + 1` or `c++` is executed. Hence, 'c' is referred to as a *counter*.

Accumulator

An accumulator is a numeric variable. When defined within a program, it keeps on changing or getting updated with the counter or some other values supplied through input. An accumulator should be initialized to avoid getting garbage value during execution of the program.

For example, s = 0;

s = s + a;

Here, the value of 's' will be updated by the input value of 'a' each time wh the statement s = s + a is executed. Hence, 's' is referred to as the *accumulat*

Shorthand operations

Java, C and C++ languages allow the use of shorthand binary operations, i.e., an expression can be written in short form as shown in the given table.

Expression	Shorthand form
a = a + b	a + = b
c = c – d	c – = d
m = m*10	m* = 10
d = d/2	d / = 2
x = x % 2	x % = 2

A shorthand expression can be written only when a variable is used as an accumulator/counter in the expression, i.e., the same variable is to be used both after and before the assignment sign.

SOLVED EXAMPLES

Example 1. If p = 5; find d = ++p + 5;

Ans.
d = ++p + 5
= 6 + 5 = 11

Example 2. If a = 48; find a = a++ + ++a; [ICSE 2006]

Ans.
a = a++ + ++a
= 48 + 50 = 98

Example 3. If c = 2; then find d = ++c + c++ + 4;

Ans.
d = ++c + c++ + 4
= 3 + 3 + 4
= 10

Example 4. If m = 12; then find n = m++ * 5 + –– m;

Ans.
n = m++*5 + --m
= 12 * 5 + (13 – 1)
= 60 + 12
= 72

Example 5. If y = 14 then find z = (++y * (y++ + 5));

Ans.
z = (++y * (y++ + 5))
= (15 * (15 + 5))
= (15 * (20))
= (15 * 20)
= 300

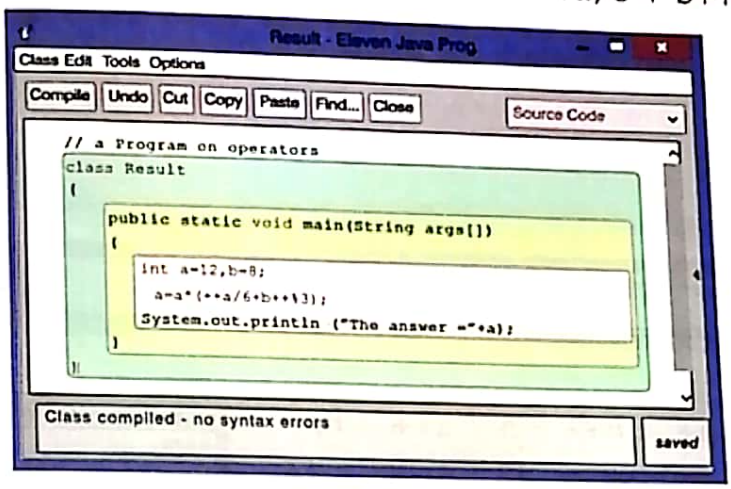
xample 6. If a = 4, b = 3; find the value of c = a++ * 6 + ++b*5 + 10;

ns.
c = a++ * 6 + ++b*5 + 10;
= 4*6 + 4*5 + 10
= 24 + 20 + 10 = 54

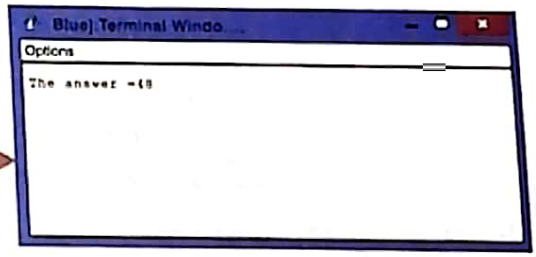
Example 7. If a = 8; find the value of a– = ++a + a++ + 4;

Ans.
```
a– = ++a + a++ + 4;
   = 8 – (9 + 9 + 4)
   = 8 – 22
   = –14
```

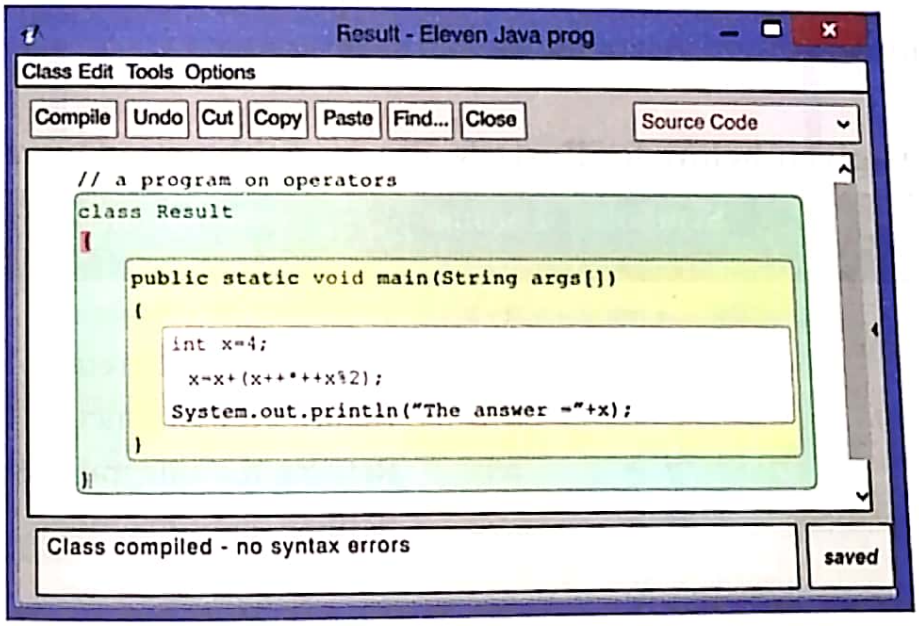
Example 8. If a = 12, b=8; find the value of a* = ++a/6 + b++ % 3;



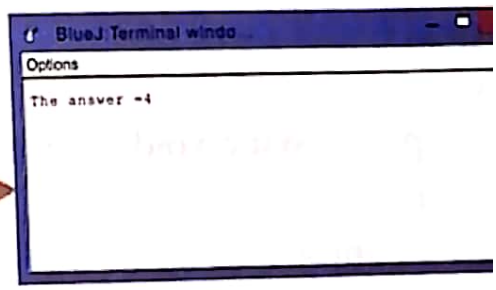
Ans.
```
a* = 12*(++a/6 + b++ % 3);
   = 12 *(13/6 + 8%3)
   = 12*(2+2)
   = 48
```



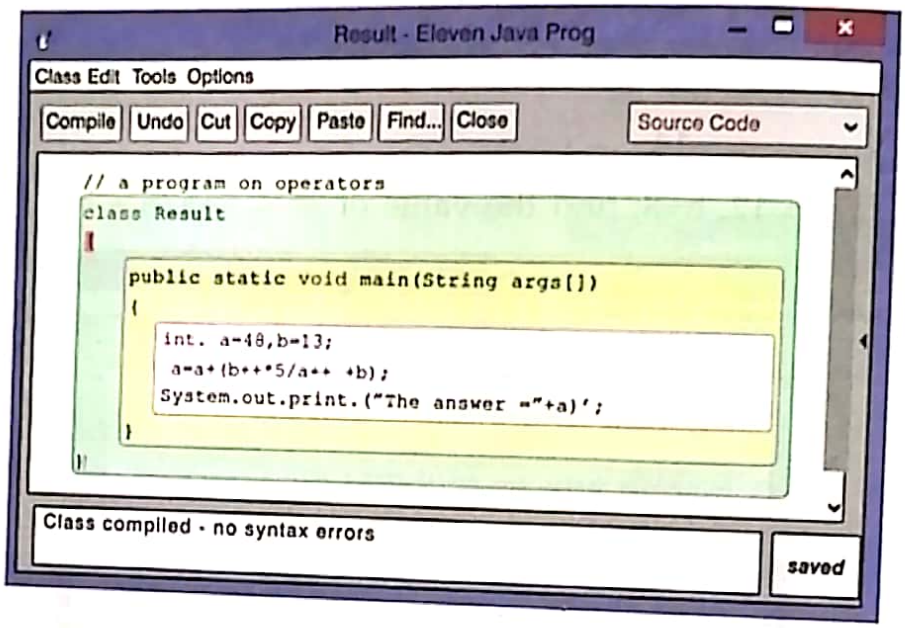
Example 9. If x = 4; find the value of x+ = x++ * ++x % 2;



Ans.
```
x+ = x++ * ++x % 2;
   = 4 + (4*6) % 2
   = 4 + (24%2)
   = 4 + 0
   = 4
```

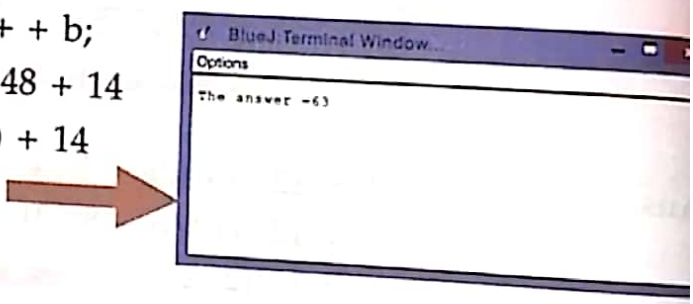


Example 10. If a = 48, b =13; find the value of a+ = b++ * 5 / a++ + b;



Ans.

a+ = b++ * 5 / a++ + b;
= 48 + (13 * 5)/48 + 14
= 48 + (65 / 48) + 14
= 48 + 1 + 14
= 63



b. Binary Operators

An arithmetic operator, which deals with two operands, is known as th *Binary Arithmetic Operator*.

For example, +, –, *, / and %, etc.

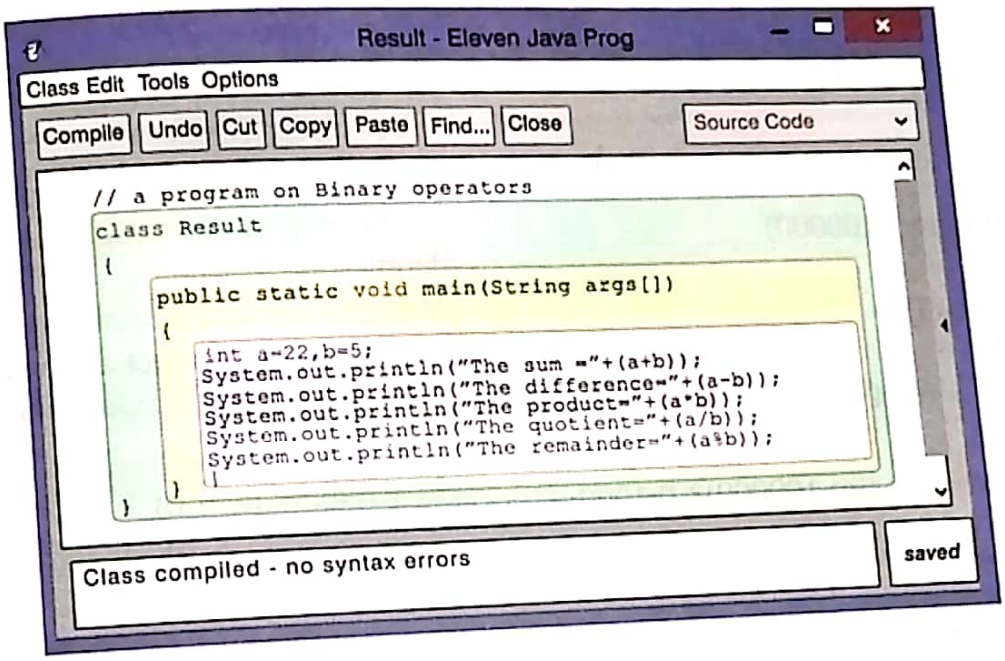
A table is given below to illustrate Binary Arithmetic Operators:

Operators	Symbols	Format	Description	Results: if a=22; b=5;
Addition	+	a+b	Returns the sum	27
Subtraction	–	a–b	Returns the difference	17
Multiplication	*	a*b	Returns the product	110
Division	/	a/b	Returns the integral part	4
Modulus/Remainder	%	a%b	Returns the remainder	2

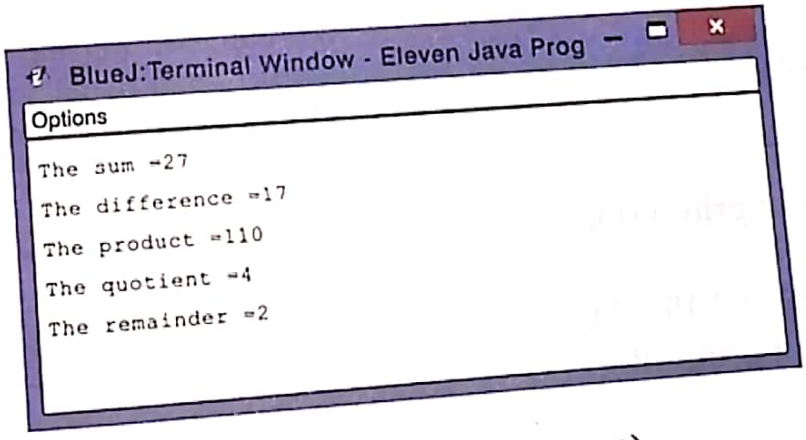
A sample program is illustrated as:

```
class Result
{
    public static void main(String args[])
    {
        int a=22,b=5;
```

```
        System.out.println("The sum="+(a+b));
        System.out.println("The difference="+(a-b));
        System.out.println("The product="+(a*b));
        System.out.println("The quotient="+(a/b));
        System.out.println("The remainder="+(a%b));
    }
}
```



The Output:



c. Ternary Operators (Conditional Assignment)

Ternary operators deal with three operands. It is also called *the condit* *assignment statement* because the value assigned to a variable depends up logical expression.

Syntax:

Variable = (test expression)? Expression 1: Expression 2.

The variable contains the expression 1 if the test condition is true, exp 2 otherwise.

For example,

a = 5; b = 3;

Max = (a>b)? a : b;
Here, the value 5 is stored in Max, as a>b is true.
Min = (b>a)? a : b;
In this case, value 3 is stored in Min, as b>a is false.

Now, rewrite the snippet using Ternary operators:

1.
```
if(a>b)
{
d = (a – b);
else
d = (b – a);
}
```
Ans. `d = (a>b)? (a – b) : (b - a);`

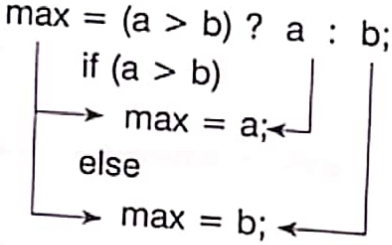
2.
```
if(basic>100000)
{
tax = 0;
else
tax = 1000;
}
```
Ans. `tax = (basic>100000)? 0:1000;`

3.
```
if((a+b)>c)
{
k = 15;
System.out.println(k);
}
if((a+b)<c)
{
k = 30;
System.out.println(k);
}
```
Ans.
```
k = ((a+b)>c)? 15 : 30;
System.out.println(k);
```

Note

The ternary operator is equivalent to the 'if-else' construct in Java language.

You can convert a ternary operator into 'if-else' construct and vice-versa by using the illustration, given below:



Further, you will learn 'if-else' construct in detail, in the chapter Conditional Construct.

Give the output of the following statements when they are executed:

(i) `int c = (3<4)? 3*4: 3+4;` Output: c = 12

(ii) `int a = 14; b = 4;`
`boolean x = (a>b)? true: false;` Output : x = true

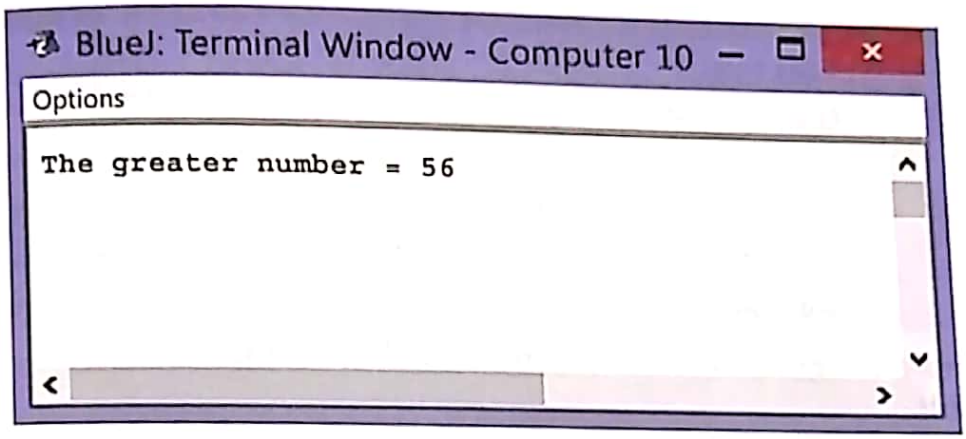
- A sample program is illustrated to find the greater of the two numbers:

```
// A program to display the greater of the two numbers
class Ternary
{

    public static void main(String args[])
```

```
    {
        int a=42,b=56,gr;
        gr=(a>b)?a:b;
        System.out.println("The greater number = "+gr);
    }
}
```

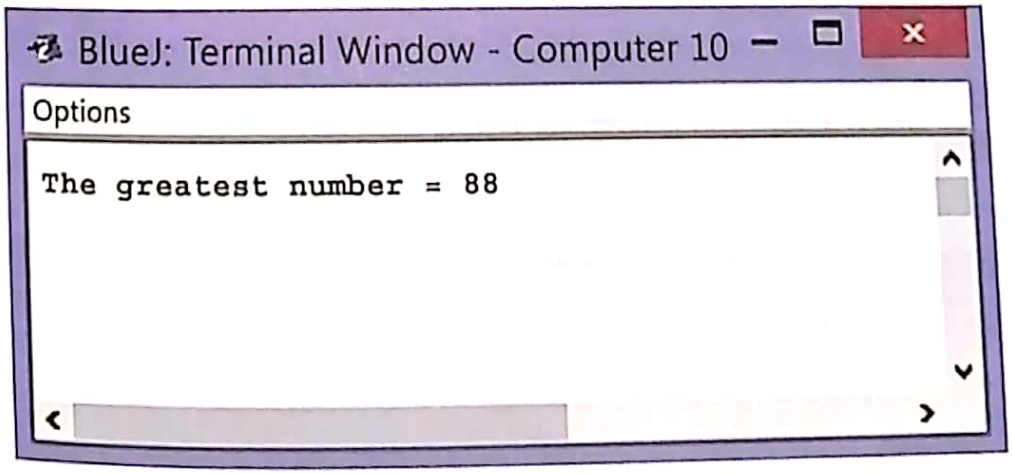
utput:



• A sample program is illustrated to find the greatest of the three numbers:

```
// A program to display the greater of the two number
class Ternary
{
    public static void main(String args[])
    {
        int a=42,b=56,c=88,max;
        max=(a>b)?(a>c?a:c):(b>c?b:c);
        System.out.println("The greatest number = "+max);
    }
}
```

)utput:



2. Relational Operators

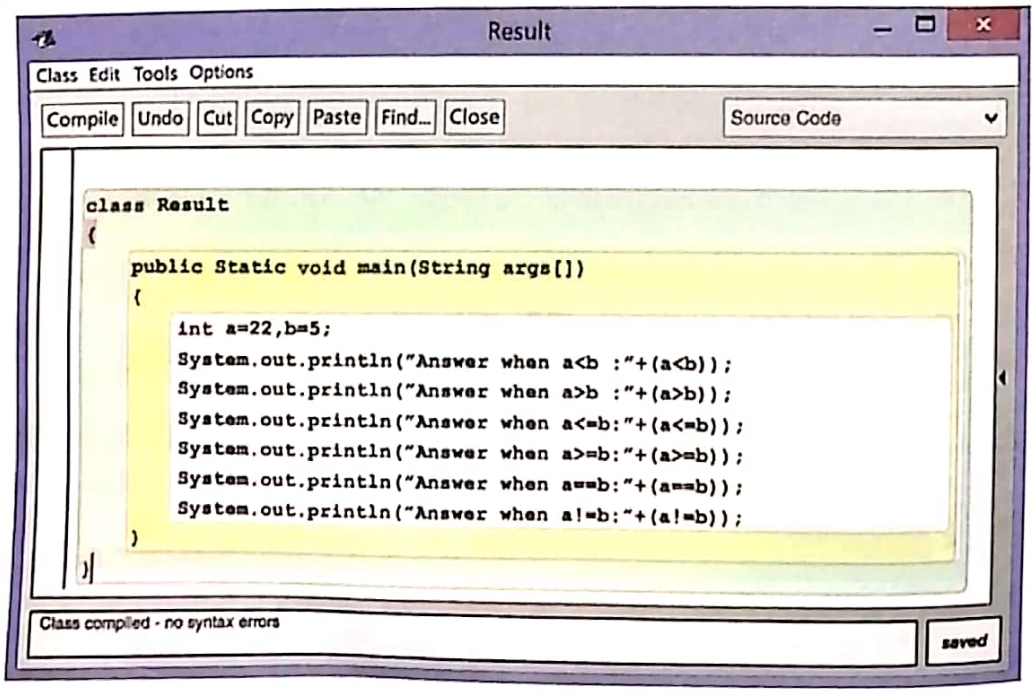
The **relational operators** compare the values of two operands and determine the relationship between them. It always returns a boolean type value (i.e. true or false). Relational operators are mainly used in control flow statements to direct the control for executing a part of the program by ignoring other parts. In order to compare numbers or characters, Java provides six types of relational operators. However, they are not applicable to compare the strings.

The different types of Relational operators are as follows:

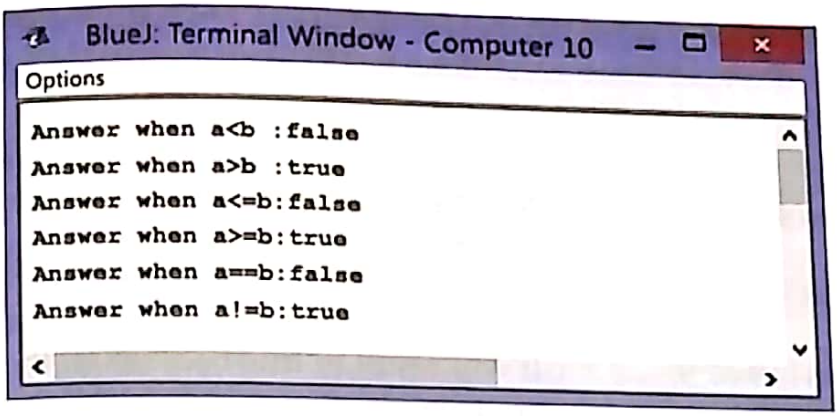
Symbol	Meaning	Format	Result if a=10;b=6;
<	Less than	a<b	false
>	Greater than	a>b	true
<=	Less than or equal to	a<=b	false
>=	Greater than or equal to	a>=b	true
==	Equal to	a==b	false
! =	Not equal to	a!=b	true

```
// A sample program to illustrate relational operator
class Result
{
    public static void main(String args[])
    {
        int a=22,b=5;
        System.out.println("Answer when a<b :"+(a<b));
        System.out.println("Answer when a>b :"+(a>b));
        System.out.println("Answer when a<=b:"+(a<=b));
        System.out.println("Answer when a>=b:"+(a>=b));
        System.out.println("Answer when a==b:"+(a==b));
        System.out.println("Answer when a!=b:"+(a!=b));
    }
}
```

Compilation and Execution:



Output:



3. Logical Operators

Java uses logical operators AND(&&), OR(||) or NOT(!). These operators yield true or false depending upon the outcome of different conditions. The different types of logical operators along with their format are shown below:

Logical operators	Symbol	Format
AND	&&	(a>b)&&(a>c)
OR	||	(a= =b)||(a= =c)
NOT	!	!(a= =b)

Precedence of logical operators is NOT (!), AND (&&) and OR (||) i.e., if a statement contains all the three logical operators then NOT is operated first.

```
boolean b = (a>b) && (b>c) || (b>a) ! (c>b)
                      ↓        ↓       ↓
                      2        3       1
```

Order of the operations is: ⟶ 1 ⟶ 2 ⟶ 3

Here, b will result in either true or false depending on the cumulative outcome of all the conditions.

Logical OR (||)

This operator is used to combine two or more conditional expressions. It will result in true if any one or more conditions (expressions) is true, otherwise in false if none of the conditions is true.

The action of logical OR (||) can be understood with the help of the following table:

Condition 1	Condition 2	Condition 1 || Condition 2
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE

It is evident from the table shown above that the result of logical OR (||) operator is FALSE only when all the conditions joined with it are FALSE. The result is TRUE if one or more conditions is TRUE.

For example,

5>4||8>12	: Results in true because 5>4 is true.
3>7||5 <=4	: Results in false because both the expressions are false.
2<0||2<12	: It will result in true because the second expression is true.
3<5||12>8||9== 9	: It will result in true because all the expressions are true.

Logical AND (&&)

The AND (&&) operator is used to combine two or more conditional expression such that it results in true if all the conditions are true, otherwise, in false.

You can understand the action of logical AND (&&) with the help of the following table:

Condition 1	Condition 2	Condition 1 && Condition 2
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

It is evident from the table shown above that the result of logical AND (&&) operator is TRUE only when all the conditions joined with it are TRUE. The result is FALSE if one or more conditions is FALSE.

For example,

5>3 && 3<5	: Results in true because both the expressions are true.
6 == 6 && 3>0	: Results in true as both the expressions are true.
5!=5 && 4 == 4	: Results in false as first expression is false.
8 > 4 && 9 >= 12	: Results in false as second expression is false

Logical NOT (!)

The Logical NOT operator is used when you want to reverse the result of conditional expression. It is a unary operator because it uses a single operand.

Let us understand the action of logical operator NOT (!) with the help of the following table:

Condition	!(Condition)
FALSE	TRUE
TRUE	FALSE

Note

Relational operators have higher precedence over logical operators AND and OR. Hence, while using AND and OR operators you need not enclose the operands within parenthesis. NOT (!) operator has the highest precedence, so it is required to enclose the operand under parenthesis.

The table shown earlier illustrates that the result of logical NOT (!) is TRUE when the condition associated with it is Flase. The result will be false when given condition is TRUE.

For example, ! (8>3) : False, because 8>3 is true.
! (5<7) : False, as 5<7 is true.
! (3<0) : True as 3<0 is false.

Indicate whether the following statements result in True or False:

(*i*) 4>7||7>15 (*ii*) 3>0||0<– 14

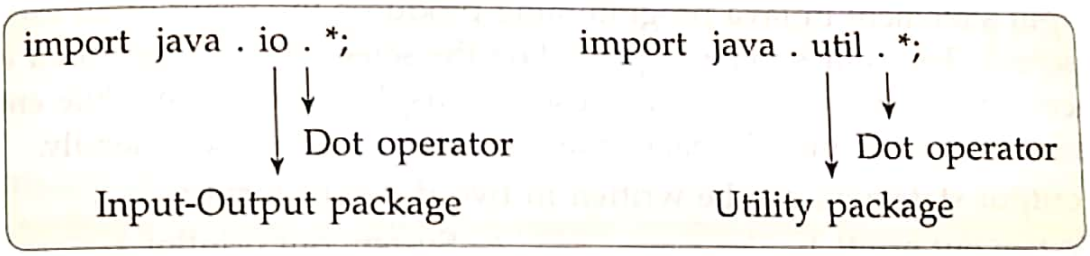
(*iii*) 5 + 3>7 && 7 >= 4 + 3 (*iv*) 3 == 4 && 5 == 5

(*v*) !(5<3)

Ans. *(i) False (ii) True (iii) True (iv) False (v) True*

The dot operator

In Java programming, you need invoking the system package to perform inpu output operation. This can be done by using the keyword 'import' to avail facilities contained in the system package. The dot operator facilitates invok members of the class to carry out the tasks. The process of importing J packages are illustrated as:



The new Operator

In the previous chapters, we already have discussed that the class representation of data and functions. When an object of a class is creat contains the data member and member methods described within the cla create an object of a class the keyword 'new' is used.

For example,

```
Result_Num nk = new Result_Num( );
```

Here, 'nk' will be an object of the class 'Result_Num'.

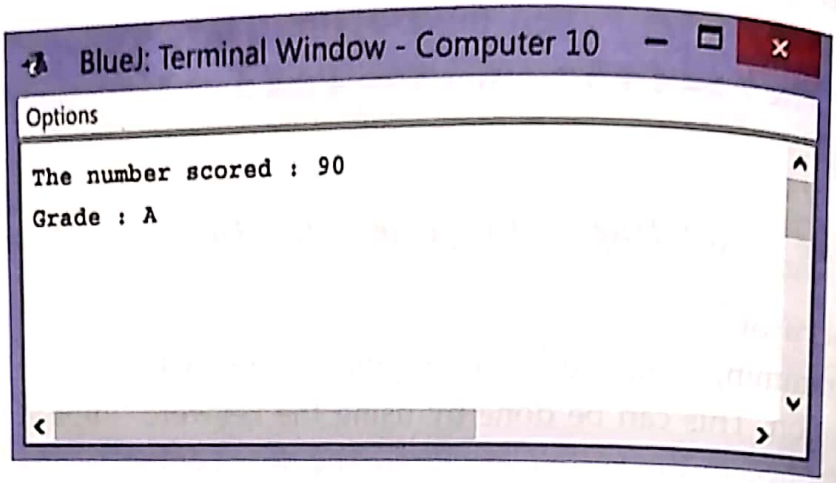
The keyword 'new' allocates space in the dynamic memory for the sto data and functions belonging to an object.

Let us understand the concept with the help of a program.

```
// A program to illustrate the concept of 'new' operator
class Result_Num
{
int num=90;
String gr="A";
public static void main(String args[])
```

```
{
Result_Num mk=new Result_Num( );
System.out.println("The number scored : "+mk.num);
System.out.println("Grade : "+mk.gr);
}
}
```

Output:



OUTPUT STATEMENT

The output statement in Java programming produces the result of an exec on the screen. The values to be displayed on the screen must be enclosed w the braces i.e. ( ). However, it is also used to display a message while ent a value into the computer. It makes the appearance more user friendly.

The output statement can be written in two different forms:

1. System.out.print( )
2. System.out.println( )

System.out.print( )

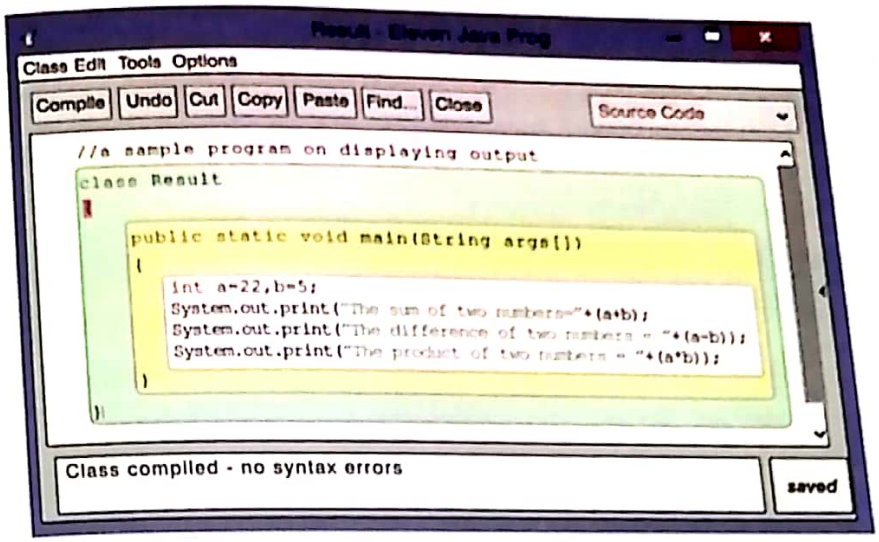
When you use this statement, the cursor remains in the same line on the screen after producing the result.

Syntax: System.out.print(<Message or value to be displayed>);

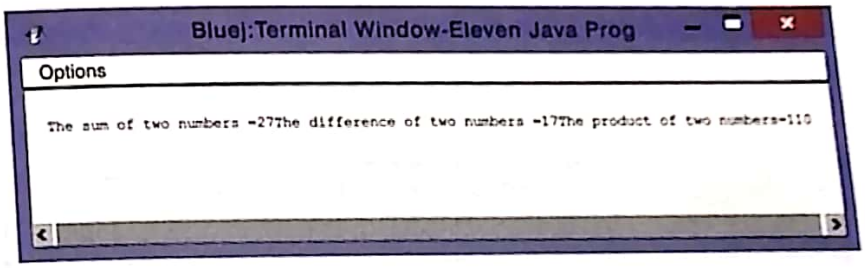
When a message is to be displaye with a variable, then they are to be used together with '+' (called connector).

A sample program is illustrated as under:

```
// a sample program on displaying output
class Result
{
    public static void main(String args[])
    {
        int a=22,b=5;
        System.out.print("The sum of two numbers = "+(a+b));
        System.out.print("The difference of two numbers = "+(a-b));
        System.out.print("The product of two numbers = "+(a*b));
    }
}
```



The Output:



System.out.println( )

This statement is used to shift the cursor to the next line of the screen after producing the result. In case, you use multiple print statements by using println() then each value will be displayed in different lines.

Hence, the word 'ln' along with 'System.out.print' acts as a line feed that will take the cursor to the next line.

Syntax: System.out.println(<Message or value to be displayed>);

The above program using System.out.println( ) statement is illustrated as:

```
// a sample program on displaying output
class Result
{
    public static void main(String args[])
    {
        int a=22,b=5;
        System.out.println("The sum of two numbers = "+(a+b));
        System.out.println("The difference of two numbers = "+(a-b));
        System.out.println("The product of two numbers = "+(a*b));
    }
}
```

REVIEW INSIGHT

(a) What is the result stored in x, after evaluating the following expression:

```
int x=5;
x=x++*2+3*--x;
```

[ICSE 2010]

Ans. x = 5*2 + 3*5
= 25

(b) What is the output of the following program snippet?

```
char x= 'A'; int m;
m=(x= = 'a')? 'A': 'a';
System.out.println("m="+m);
```

[ICSE 2010]

Ans. m = 97

(c) What is the difference between / and % operator? [ICSE 2011]

Ans. The '/' operator results in the quotient (integer part) when a number is divided by another number.

For example, int a=3,b=7,c;

c=b/a;

The output: c= 2

The '%' operator results in remainder when one number is divided by other.

For example, int a=3,b=7,c;

c=b%a;

The output: c= 1

(d) What will be the output of the following code?

```
int k=5,j=9;
k+= k++ - ++j + k;
System.out.println("k="+k);
System.out.println("j="+j);
```

[ICSE 2011]

Ans. k=6
j=10

(e) Write an expression in Java for

$z = 5x^3 + 2yx + y$

[ICSE 2011]

Ans. z = 5*x*x*x + 2*y*x + y

(f) Give one point of difference between the Unary and the Binary operators. [ICSE 2012]

Ans. The unary operator is used to perform operation on single operand.

For example, ++, --, etc.

The binary operator performs operation on two operands. e.g., +, -, *, / and %.

(g) What are the values of x and y when the following statements are executed?

```
int a=63, b=36;
boolean x=(a>b)?a:b;
int y=(a<b)?a:b;
```

[ICSE 2012]

Ans. The statement given below is wrong. It will produce an error 'Incompatible type

```
boolean x=(a>b)?a:b;
```

It should be :

```
boolean x=(a>b)?true : false; then,
x = true
y = 36
```

}

HIERARCHY OF OPERATORS

The computer uses precedence of binary operator
In mathematics, we use the BODMAS System of H
we use BEDMAS.

Order of Operation

BRACKET
EXPONENT
DIVISION/ MULTIPLICATION
ADDITION/ SUBTRACTION

Complete precedence of operators at a glanc

Operators	Hierarchy
( ) , [ ]	1
++, - -, ~, !	2
*, /, %	3
+, -	4
>>, >>>, <<	5
>, >=, <, <=	6
= =, !=	7
&	8
^	9
!	10
&&	11
||	12
?=	13
=	14

Mul
hav
This
whi
left
be
it a
sub

(h) What will be the result stored in x after evaluating the following expression? [ICSE 2012]

```
int x=4;
x+=(x++)+(++x)+x;
```

Ans. x = 20

(i) What do you mean by precedence of operators? [ICSE 2013]

Ans. The hierarchical order in which the operators are used for operation is known as precedence of operator.

For example,

Precedence of arithmetical operators : Brackets, Exponent, Multiplication/ Division, Addition/Subtraction.

Precedence of logical operators : Not, and, or (!, && and ||)

(j) Rewrite the following program segment using if....else statement. [ICSE 2013]

```
comm = (sale>15000) ? sale*5/100 : 0;
```

Ans. Snippet by using if....else statement:

```
if(sale > 15000)
comm = sale * 5/100;
else
comm = 0;
```

(k) Write a Java expression for $ut + \frac{1}{2}at^2$. [ICSE 2013]

Ans. Java expression for the given mathematical expression:

```
u * t + 1.0/2.0 * a * t* t
```

(l) Operators with higher precedence are evaluated before operators with relative lower precedence. Arrange the operators given below in order of highe precedence to lower precedence.

(*i*) && (*ii*) %

(*iii*) >= (*iv*) ++ [ICSE 201

Ans. The correct order of operators is:

(*i*) ++ (*ii*) %

(*iii*) >= (*iv*) &&

(m) Rewrite the following program segment using if-else statements instead of t ternary operator.

```
String grade = (marks>=90)?"A": (marks>=80)? "B": "C";
```

[ICSE 201

Ans.

```
if(marks>= 90)
grade = "A";
else
if(marks>=80)
grade= "B";
else
grade= "C";
```

(n) Give the output of the following method:

```
public static void main(String[] args)
{
int a = 5;
a++;
System.out.println(a);
a -= (a--) - (-- a);
System.out.println(a);
}
```

[ICSE 201

Ans. 6 and 4

(o) Evaluate the value of n, if value of p=5, q=19
int n = (q-p) > (p-q)? (q-p): (p-q); [ICSE 2015]

Ans. n =14

(p) Write the Java expression for:
$\frac{a^2 + b^2}{2ab}$ [ICSE 2015]

Ans. (a*a + b*b)/2.0*a*b

(q) If int y =10 then find int z = (++y*(y+++5)); [ICSE 2015]

Ans. z = 11*(11+5)
= 176

(r) Write down Java expression for:
$t = \sqrt{a^2 + b^2 + c^2}$ [ICSE 2016]

Ans. t= Math.sqrt (a*a+ b*b+ c*c)

(s) Rewrite the following using ternary operator: [ICSE 2016]
if (x%2 = = 0)
System.out.println("Even");
else
System.out.println("Odd");

Ans. (x%2==0)? System.out.println("Even"):System.out.println("Odd");

(t) Give the output of the following expression: [ICSE 2016]
a+= a++ + ++a + --a + a--; when a = 7;

Ans. 7 + (7 + 9 + 8 + 8) = 39

Student's Notes