COMPUTER SCIENCE CLASS X

CHAPTER 1(Unit I to VI) Revision of Class IX Syllabus

Some Key Points

DATA ABSTRACTION (In simple words)

- Hiding unnecessary details.
- Knowing necessary features.

Let us consider one real life example to understand more about abstraction-

Let's consider a car. A car is actually a very complicated machine. Every car is made up of thousands of individual parts each of which serves a different purpose. But while buying a car we do not go inside the engineering details but only consider the necessary ones such as how to drive, how to apply brake and so on…

- **Abstract class:** is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).
- **Abstract method:** can only be used in an abstract class, and it does not have a body. The body is provided by the subclass (inherited from).

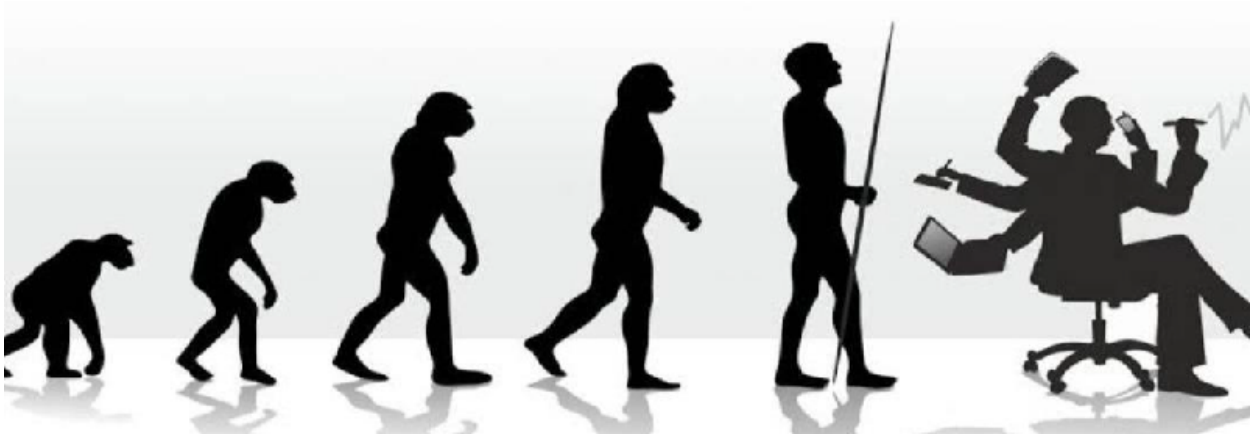
(An abstract class can have both abstract and regular methods)

Ex:-

```
abstract class Animal
{
public abstract void animalSound() ;
public void sleep()
{
System.out.println("Zzz");
}
}
```

INHERITANCE

Let's take the following example to understand inheritance:



We all know that humans have evolved through ages, we also know that our ancestors were probably monkey or chimpanzees… but in today's world are we considering ourselves to the monkey class only?? Of course not… therefore we can say that monkey was a class but human have evolved so much that they have now become a totally new class with only some characteristics as the monkey class.
Thus, we can conclude that **taking, suppressing or adding some characteristics or behaviors from another class and making or defining a whole new class is inheritance.**
Here, from the above real life example we can say that **monkey class is the base class or super class** and **human class is the derived class or sub-class.**

- **Base class**: the class that gets inherited to another class.
- **Derived class**: the class that inherits from the base class.
- **Reusability**: using an existing code to solve a new problem.

The keyword used for inheritance is **extends**.
Syntax :

```
class derived-class extends base-class
{
//methods and fields
```

}

POLYMORPHISM

· The word polymorphism means having **many forms.**

• In simple words, we can define polymorphism as the process of using a method or function for more than one purpose.

• Polymorphism is considered as one of the important features of Object Oriented Programming.

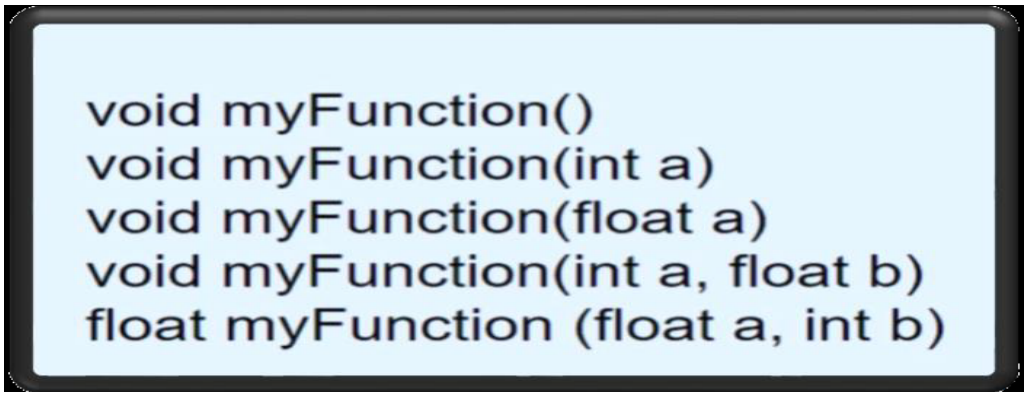
• It is implemented by using **function overloading.**

Real life example of polymorphism

A person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee. So the same person posses different behavior in different situations. This is called polymorphism.

Function overloading :

When there are multiple functions with same name but different parameters then these functions are said to be overloaded.



Here, in the above example you can see that the method name or the function name is same i.e “myFunction” but the parameters for all the above methods are different. And this is how it can be understood that which method is to be called.

ENCAPSULATION

In simple words. the action of enclosing something in or as if in a capsule. **Encapsulation in Java** is a process of wrapping code and data together into a single unit, for example, a capsule which is mixed of several medicines. Bundling similar fields and methods inside a class together also helps in **data hiding.**

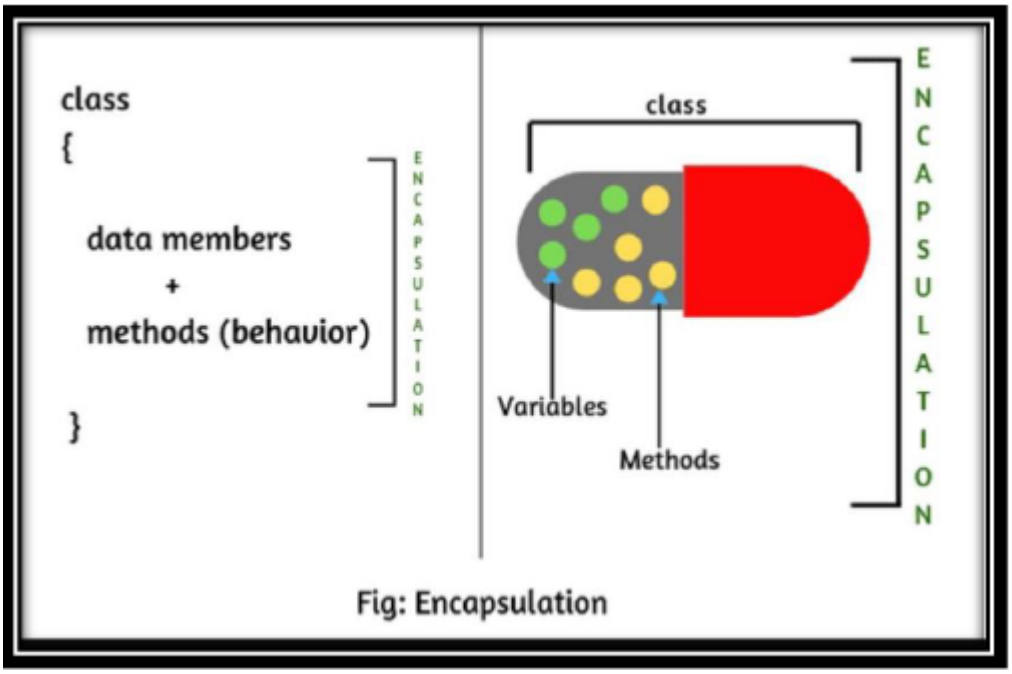


Fig: Encapsulation

As shown in the above figure:

The data members and the methods are together kept in a class and hence we can say that they are encapsulated within the class.

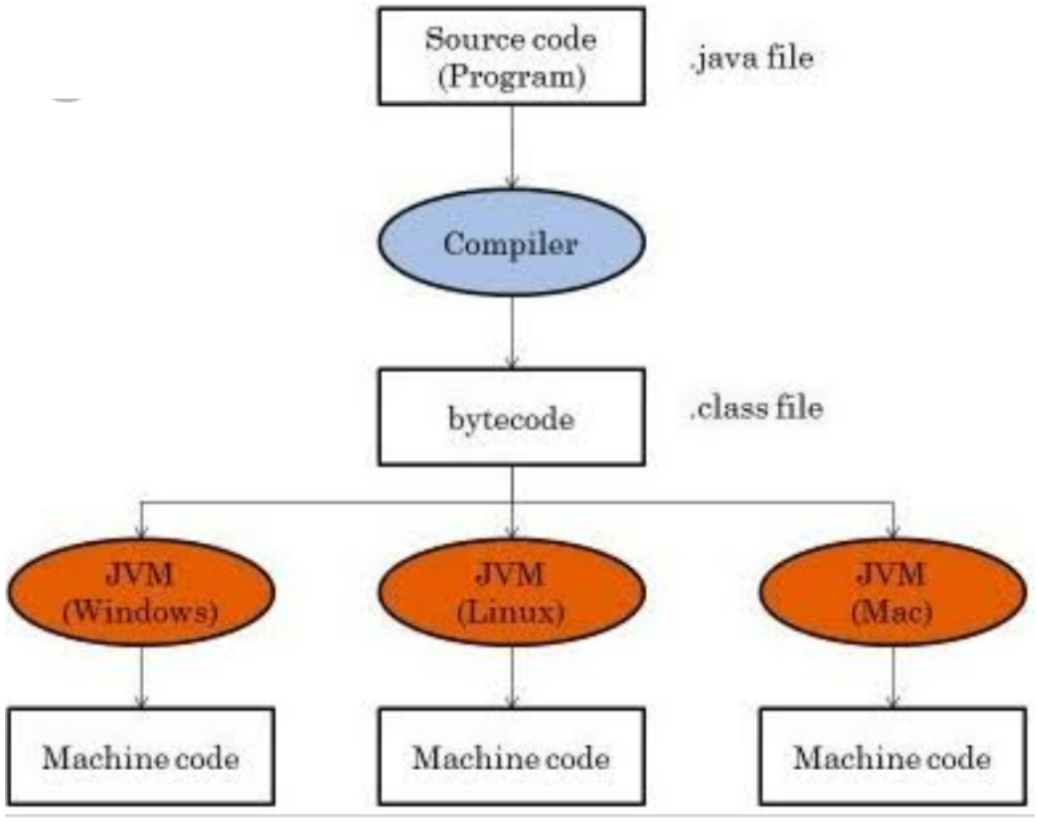
Advantages of encapsulation:

• The ***variables in Java*** or the method of the class are hidden from any other class and cannot be accessed outside the class.
• We can also call it, as Data-hiding.
• The encapsulated class is easy to test.
• It can be achieved by declaring the class as private while the methods as public so that the variables can be accessed.

Basic Features of java:

1. Java is a object oriented programming language.
2. Java programs are both compiled and interpreted.
3. It can access data from local system as well as from net.
4. Java doesn't require any preprocessor(#) or inclusion of header files for creating a java program.
5. Java is case sensitive language.

Compiler and Interpreter



Java package:

Java package is a collection of various classes which can be included in our program with the help of import keyword.

Java.lang is the default package of java programming.

Some example of java packages are: java.io, java.applet, java.net, java.math etc.

Java Reserved words:

Java reserved words or keywords are those words which carry special meaning to the java compiler. It can not be used as variables, methods, classes or any other identifiers.

Some of the reserved words or keywords are:

case	switch	else	break
static	main	do	float
For	double	Public	int
new	import	class	default

Different types of java programming :

1. Java application
2. Java applet or Internet applet

Output Statement in Java Programming:

System.out.print()-After printing(displaying the result in output screen) cursor remain in the same line.

System.out.println()-After printing(displaying the result in output screen) cursor goes to the next line.

Primitive Data Types With their Sizes:

Primitive Data Types with their sizes and the ranges at a glance:

Data Type	Size	Range	Description
byte	8 bits (1 byte)	–128 to +127	Bit-wise operations
short	16 bits (2 bytes)	–32768 to +32767	To represent short integers
int	32 bits (4 bytes)	–231 to 231 – 1	To represent integers

S

Java Operator Precedence Table

Precedence	Operator	Type	Associativity
15	() [] .	Parentheses Array subscript Member selection	Left to Right
14	++ --	Unary post-increment Unary post-decrement	Right to left
13	++ -- + - ! ~ (*type*)	Unary pre-increment Unary pre-decrement Unary plus Unary minus Unary logical negation Unary bitwise complement Unary type cast	Right to left
12	* / %	Multiplication Division Modulus	Left to right
11	+ -	Addition Subtraction	Left to right
10	<< >> >>>	Bitwise left shift Bitwise right shift with sign extension Bitwise right shift with zero extension	Left to right
9	< <= > >= instanceof	Relational less than Relational less than or equal Relational greater than Relational greater than or equal Type comparison (objects only)	Left to right
8	== !=	Relational is equal to Relational is not equal to	Left to right
7	&	Bitwise AND	Left to right
6	^	Bitwise exclusive OR	Left to right
5	|	Bitwise inclusive OR	Left to right
4	&&	Logical AND	Left to right
3	||	Logical OR	Left to right
2	? :	Ternary conditional	Right to left
1	= += -= *= /= %=	Assignment Addition assignment Subtraction assignment Multiplication assignment Division assignment Modulus assignment	Right to left

Larger number means higher precedence.

Difference between Testing and Debugging:

Testing	Debugging
1. Process of finding and locating defects of the software	1. Process of fixing the identified defects.
2. Purpose is to find many defects as possible	2. Purpose is to remove the detected defects.
3. Can be done manually or automatically	3. Done manually.

Mathematical Library Methods

Java provides many useful methods in the Math class for performing common mathematical functions.
All mathematical functions are included in a class called "Math" that is part of java.lang package(a default package).
Most methods of the Math class are static and therefore there is no need to instantiate any new object of the Math class before using it.

e.g. Math.<Method name> i.e. Math.sqrt(parameter)

Math.min()

This function returns the minimum of two numbers. The return value depends on the values used as the arguments of the function (i.e., if the arguments are integer numbers, the result are also integer numbers(int data type) and so on.

e.g.
int n = Math.min(4,6); returns 4
double b=Math.min(4,6.2) returns 4.0

Math.max()

This function returns the maximum of two numbers. The return value depends on the values used as the arguments of the function (i.e., if the

arguments are integer numbers, the result are also integer numbers(int data type) and so on.

e.g.

int n = Math.max(4,6); returns 6
double b=Math.min(4.2 , 6); returns 6.0

Math.pow(a,b)

Returns the numbers a raised to power b. It always returns its value in double data type. 'a' is called base and 'b' is called the index.

e.g.

double d=Math.pow(2,3); returns $2^3 = 8.0$
double d= Math.pow(2.0,3.0); returns $2.0^{3.0} = 8.0$
double d = Math.pow(5.0,-2.0); returns 0.04

Math.sqrt()

This function is used to find the square root of a positive number. It always returns a double data type value.

e.g.

double d = Math.sqrt(16) returns 4.0
double d = Math.sqrt(25.0) returns 5.0

Math.cbrt()

This function is used to find the cube root of a positive or a negative number. It always returns a double data type value.

e.g.

double d= Math.cbrt(125); returns 5.0
double d= Math.cbrt(-3.375); returns -1.5

Math.log()

This function returns the natural logarithmic value of a given argument. It always returns a double data type value.

e.g.

double d= Math.log(6.25) return 1.8325

Math.abs()

This function is used to return the absolute value of an argument(i.e., magnitude of the number without its sign i.e., a positive value). It returns int/long/double data type value depending upon the arguments supplied.

e.g.

```
int n = Math.abs(3);          returns 3
int n = Math.abs(-8);         returns 8
double d = Math.abs(3.66);    returns 3.66
double d= Math.abs(-9.99);    returns 9.99
```

Math.floor()

This function returns the largest integer which is less than or equal to the argument. It always returns a double data type value whether the argument is integer(int), float or a double data type.

e.g.

```
Math.floor(5);          returns   5.0
Math.floor(5.892)       returns   5.0
Math.floor(5.5)         returns   5.0
Math.floor(-0.48)       returns   -1.0
Math.floor(-5.892)      returns   -6.0
```

Math.ceil()

This function returns the smallest integer which is greater than or equal to the argument. It always returns a double data type value whether the argument is integer (int), float or a double data type.

e.g.

```
Math.ceil(5);           returns   5.0
Math.ceil(5.892)        returns   6.0
Math.ceil(5.5)          returns   6.0
Math.ceil(-0.48)        returns   -0.0
Math.ceil(-5.892)       returns   -5.0
```

Math.round()

This function returns the value in rounded-off form. This function always returns the integer value in long or int data type. It predicts the different outputs for positive and negative numbers.

For Positive Numbers:

If fractional part lies between 0(inclusive) and 0.5(exclusive)then the function Math.round() returns the integer part of the number.

e.g.

Math.round(8.0); returns 8
Math.round(8.49); returns 8

If the fractional part is 0.5 or more then the function returns the next higher integer of the number.

e.g.

Math.round(8.5); returns 9
Math.round(8.99); returns 9

For Negative Numbers:

If fractional part lies between 0(inclusive) and 0.5(inclusive) then the function Math.round () returns the integer part of the number.

e.g.

Math.round(-8.0); returns -8
Math.round(-8.5); returns -8

If the fractional part is more than 0.5, then the function returns the next lower integer of the number.

e.g.

Math.round(-8.51); returns -9
Math.round(-8.99); returns -9

Math.rint()

It returns the truncated value (integral part) of a number. It always returns integer value in double data type. It predicts different output for positive and negative numbers.

For Positive Numbers:

If fractional part lies between 0(inclusive) and 0.5(inclusive)then the function Math.rint() returns the integer part of the number.

e.g.

Math.rint(8.0); returns 8.0
Math.rint(8.5); returns 8.0

If the fractional part is more than 0.5, then the function returns the next higher integer of the number.

e.g.

Math.rint(8.501); returns 9.0
Math.rint(8.99); returns 9.0

For Negative Numbers:

If fractional part lies between 0(inclusive) and 0.5(exclusive) then the function Math.rint () returns the integer part of the number.

e.g.

Math.rint(-8.0); returns -8.0
Math.rint(-8.49); returns -8.0

If the fractional part is 0.5 or more , then the function returns the next lower integer of the number.

e.g.

Math.rint(-8.5); returns -9.0
Math.rint(-8.99); returns -9.0

Math.random()

Math.random() returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0.

The random function can be used to obtain the result 'head' or 'tail' randomly when a coin tossed. The function can be used as:

```
int  n = (int) (Math.random()*2);
```

The value of n (i.e.; 0 or 1) so obtained be assumed as 'tail' or 'head'

respectively.

Answer the following questions in your copy….

1. What does reusability mean?
2. Which OOP principle implements function overloading.
3. What is the use of keyword ‘import’?
4. Define Byte code and JVM.
5. Distinguish between
    a) Compiler and Interpreter.
    b) Source code and object code
    c) Object and class
6. Define a) Java application b) Internet applet
7. What is token ?explain with an example.
8. What is constant or literal? Explain different types of constant.
9. Explain static initialization and dynamic initialization with an example.
10. A class is also referred as ‘object factory’? Comment.
11. What is the significance of using the word ‘new’ while creating an object?
12. Write a java statement to create an object mp4 of class digital.
13. What are the rules to naming a variable in a Java programming?
14. Explain the term ‘type casting’?
15. Define the following with an example each:
    a) Implicit type conversion

b) Explicit type conversion

16. Why is an object called an ‘Instance’ of a class? Explain.
17. What do you understand by primitive data type? Give examples.
18. What do you understand by non-primitive data type? Give examples.
19. Distinguish between:

    a) Character and string

    b) Integer and floating constant

20. Differentiate between an operator and an expression.
21. What is operator? Explain different types of operator.
22. What is ternary operator? Explain with the help of an example.
23. What is the difference between / and % operator? Give suitable example.
24. Distinguish between:

    a) = and ==

    b) a+1 and a++

    c) (p!=q) and !(p==q)

    d) true and “true”

25. Give one point of difference between unary and binary operators?
26. What do you mean by precedence of operators?
27. What is compound statement? Explain with an example.
28. Explain different types of error with an example.
29. What is the difference between the scanner class element next() and

nextline()?

30. What will the final values be stored in variables when executed?

```
double a = -99.51;
double b = -56.25;
```

i) `double p = Math.abs(Math.floor(a));`

ii) `double q = Math.sqrt(Math.abs(b));`

iii) `double r = Math.rint(Math.abs(b));`

31. Rewrite the snippet using Ternary Operators:

```
if(a<b)
 c=a+b;
 else
 c=a-b;
```

32. Give the output of the following expression:

a) a+= a++ + ++a + --a + a—when a=7

b) int a=6, b=5,c; c=(a++ % b++) * a + ++a*b++;

c) int a=6,b=5; a+= a++ % b++ *a + b++ * --b;

33. Write Java Expression for the following:

a) $\frac{\sqrt{3}}{4}a^2$ b) $\frac{(a+b)^2}{ab}$ c) $a^2 + b^3 + c^4$ d) $\sqrt[3]{mn} + mn^3$

*****End*****

****** Children are requested to go through the notes thoroughly and write down the notes and the questions and their answers in your respective copies. All the subject copies will be checked as soon as the school reopens******

Book: Understanding Computer Applications with BlueJ ICSE Class X

Chapter 1: Unit I to Unit VI

Teacher: Mr.Sabyasachi Mukherjee **Date**: 01/05/2020