**Q : What is file in computer system?**

A: A file is an object which stores data or information, setting or commands with a computer program. It is stored in a storage device. There are several types of file: docs, xls, txt, DAT, mp3, mp4 etc.

**Q: What is class?**

A: A class is an entity that determines how an object will behave and what the object will contain. In other words, a class in a blueprint or set of instructions to build a specific type of object. A class is like a repository which contains all data members and methods.

Ex:

Class <class\_name> {

Fields;

Methods;

}

**Q: What is an object?**

A: An object is an instance of a class. It is nothing but a self-contained components which consists of methods and variables to make a particular type of data useful. Object determines the behavior of the class. When you send a message to an object, we are asking the object to execute one of its method. An object is an instance of a class which acquires all the properties and behavior of the class.

Ex:

ClassName ReferenceVariable = new ClassName();

**Q. What is constructor?**

A: A constructor is a special type of method in a class which is called automatically when the object of the class is created.

To write a constructor, there are two rules:

1. Class name and Constructor name should be exactly same.
2. It should not have any return type.

There are two types of constructor Default & Parameterized constructor.

**Q. What are the different OOP principles?**

A: There are basically 4 types of principles in OOP.

1. Encapsulation
2. Inheritance
3. Polymorphism
4. Abstraction

**Encapsulation:**

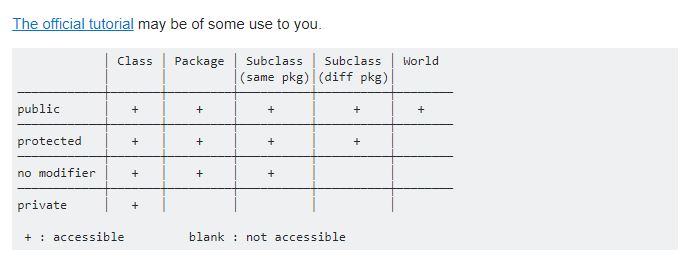
Encapsulation is the mechanism of data hiding from other classes that keeps data and functions that manipulate data safe from outside interference and misuse. Hiding internal state and requiring all interaction to be performed through an object method is called data encapsulation.

Encapsulation is:

* Binding the data with the code that manipulates it.
* It keeps the data and the code safe from external interference

OOP has 3 types of data access modifier:

1. Public
2. Protected
3. Private



Inheritance:

Inheritance is a property in which the property of a parent class (super-class) is passed on to child class (sub-class).

Polymorphism:

So polymorphism is the ability (in programming) to present the same interface for differing underlying forms (data types).

**Q. What is instance variable?**

A: Instance variable is the variable which declared inside a class but outside of method, constructor or any other block. Variables which are declared without static keyword are known as instance variable.

* Instance variables are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed.
* class IronMan{
* /\*\* These are all instance variables \*\*/
* public String realName;
* public String[] superPowers;
* public int age;
* /\*\* Getters / setters here \*\*/
* }

**Q. What is class/static variable?**

A: Static variable in java is variable which belongs to the class and initialized only once at the start of the execution.

* It is a variable which belongs to the class and not to object(instance)
* Static variables are initialized only once, at the start of the execution. These variables will be initialized first, before the initialization of any instance variables
* A single copy to be shared by all instances of the class
* A static variable can be accessed directly by the class name and doesn’t need any object

class VariableDemo

{

Public static int count=0;

}

**Q. What is singleton class?**

A. Singleton class is a class which have only one instance at a time. After first time if we try to instantiate the singleton class, the new variable will point the first instance created. So whatever modification we do to any variable inside the class through any instance, it affects the variable of the single instance created.

To create a singleton class:

1. Make constructor as private
2. Write a method that has return type object of this singleton class. Here the concept lazy initialization is used to write this static method.

|  |
| --- |
| 1. class Singleton 2. { 3. // static variable single\_instance of type Singleton 4. private static Singleton single\_instance = null; 6. // variable of type String 7. public String s; 9. // private constructor restricted to this class itself 10. private Singleton() 11. { 12. s = "Hello I am a string part of Singleton class"; 13. } 15. // static method to create instance of Singleton class 16. public static Singleton getInstance() 17. { 18. if (single\_instance == null) 19. single\_instance = new Singleton(); 21. return single\_instance; 22. } 23. } 25. // Driver Class 26. class Main 27. { 28. public static void main(String args[]) 29. { 30. // instantiating Singleton class with variable x 31. Singleton x = Singleton.getInstance(); 33. // instantiating Singleton class with variable y 34. Singleton y = Singleton.getInstance(); 36. // instantiating Singleton class with variable z 37. Singleton z = Singleton.getInstance(); 39. // changing variable of instance x 40. x.s = (x.s).toUpperCase(); 42. System.out.println("String from x is " + x.s); 43. System.out.println("String from y is " + y.s); 44. System.out.println("String from z is " + z.s); 45. System.out.println("\n"); 47. // changing variable of instance z 48. z.s = (z.s).toLowerCase(); 50. System.out.println("String from x is " + x.s); 51. System.out.println("String from y is " + y.s); 52. System.out.println("String from z is " + z.s); 53. } 54. } |

Output:

String from x is HELLO I AM A STRING PART OF SINGLETON CLASS

String from y is HELLO I AM A STRING PART OF SINGLETON CLASS

String from z is HELLO I AM A STRING PART OF SINGLETON CLASS

String from x is hello i am a string part of singleton class

String from y is hello i am a string part of singleton class

String from z is hello i am a string part of singleton class

**Q. What is lazy initialization?**

A. Lazy initialization is the memory conservation technique in OOP. It actually acts like cache. If we have a private member in a class that has been already initiated and again try to initiate the same member, then we return the already created instance. This technique is called initialization.

Ex: Upper code… (line 20, 21)

**Q. What is “final” class?**

A. Final class is simply a class which can’t be extended. It is immutable and can’t be inherited. If we want to prohibit a class from inheritance then we can declare that class as final class. Please remember a final class can still extend another non-final class. But a final class can’t be extended to another class.

* A final class cannot be extended by any other class
* A final variable cannot be reassigned another value
* A final method cannot be overridden

Some of the most well-known final classes in Java are the String, Math, and the various Wrapper classes.

Public final class Car() {

}

**Q. What is multiple inheritance in Java?**

A. Java doesn’t support multiple inheritance in classes. Because it can lead Diamond Problem. Rather than it support multiple inheritance for interfaces.

<https://www.geeksforgeeks.org/java-and-multiple-inheritance/>

**Q. What is interface in Java?**

A. An interface is a blueprint of a class. It is just like java class, but it has only static constant and abstract method. Java uses interface to implement multiple inheritance. All methods in interface are implicitly public and abstract.

Interface printable {

Int min = 5;

Void print()

}

If we complete this interface then it turns into:

Interface printable {

Public static int min = 5

Public abstract void print();

}

A class must implement an interface’s method (if inherit).

**Q. Differentiate between class and interface?**

A. Below are the difference between interface and class:-

* The interface cannot be instantiated.
* An interface doesn’t have any constructors.
* Interface only have abstract methods.
* A class implements an interface and extends a class.
* An interface can extend multiple interfaces.

**Q. What is abstract class?**

A. A class which contains the abstract keyword in declaration is called abstract class. It can have abstract or non-abstract methods. It needs to be extended and its method implemented. It can’t be instantiated.

* An abstract class must be declared with an abstract keyword.
* It can have abstract and non-abstract methods.
* It cannot be instantiated.
* It can have constructors and static methods also.
* It can have final methods which will force the subclass not to change the body of the method.
* Abstract classes may or may not contain abstract methods but, if a class has at least one abstract method, then it must be declared abstract.
* The abstract class cannot be instantiated.
* To use an abstract class, we have to inherit it from another class.
* If we inherit an abstract class, then we have to provide implementations to all the abstract methods in it.