Question Set: 1

Oops Concept Interview questions

1. What are the four pillars of oops concept?

Answer-:

The four pillars of oops concept are:

- Inheritance
- Polymorphism
- Encapsulation
- Abstraction

Inheritance:

i. Here we inherit the parent class members to the child class with the intention of reusing them in child class.
ii. The main advantage of inheritance is reusability of code.
iii. We use extends keyword for class to class and interface to interface inheritance, If we want to inherit interface to class then we will use implements keyword.
iv. We can't inherit Static methods, Static variables and constructors.
v. Private members can't be inherited.

Polymorphism:

i. Poly means many and morphism means form, here we develop a feature in such a way that it can take more than one form according to our requirement.
ii. There are two types of polymorphism:
    - Overriding
    - Overloading

Overriding:

a) For overriding to happen inheritance is mandatory without inheritance overriding is not possible.
b) Here first we inherit the parent class members to child class and then we change the logic of the inherited method in child class as per our

requirement by keeping the signature of the method same as parent class method.

c) We can check if overriding is done or note by using @Override annotation.

Overloading:

a. Inheritance is not mandatory for method overloading.
b. Here we create more than one method inside the same class by keeping All the method signature same provided different numbers of arguments or different types of arguments.
c. Static method can be overloaded.
d. Overloading increases the readability of the code.

Encapsulation:

i. Here we wrap the data members with the help of methods so that direct access is not possible to the data members and the methods can only be operate on the data members.
ii. Here we make the variables private and wrap the variables with the help of getter and setter methods.
iii. Encapsulation is the best example of data hiding.

Abstraction:

i. Hiding the implementation details is known as abstraction.
ii. We can achieve abstraction with the help of interface and abstract class.

2. <u>What is interface and why we use interface before 1.8?</u>

i. Interface in java is like a class but the main difference is we can store only abstract methods in it.
ii. We can't keep static methods in an interface.
iii. Interface supports multiple inheritance.
iv. Variables which are declared in an interface is by default public static final.
v. An interface can be public and default.
vi. We use interface to achieve abstraction.
vii. By using interface we can achieve 100% abstraction.

3. <u>What is abstract class in java?</u>

i. We can create an abstract class by using abstract keyword.
ii. We can create both complete and incomplete methods inside an abstract class.
iii. Abstract class can have static methods.
iv. Methods in an abstract class can't be final because if we make a method final then we can't override them and methods in abstract class are meant for overriding.
v. We can't create object for an abstract class.
vi. An abstract class can have constructor.
vii. By using an abstract class we can achieve abstraction but the abstraction can be from o to 100%.

4. <u>If we are not allowed to create an object for abstract classes then why we can create constructor in abstract class?</u>

i. An abstract class can have instance variable and no abstract methods so to initialize them constructor concept is there for abstract class.

5. <u>Why multiple inheritance is not present in java?</u>

i. At class level java don't support multiple inheritance but at interface level java supports multiple inheritance.
ii. To prevent the ambiguity at class level this concept is not present.

6. <u>What is object cloning and what is the use of object cloning?</u>

   i. In java object cloning means creating an exact copy of an object.
   ii. We can do object cloning by using the clone() method which is present in object class.
   iii. For object cloning implementing the clonable interface is mandatory.
   iv. If an object have two references then if we make any changes in one reference it will effect the object and change the object behavior so to avoid that object cloning is required.

7. <u>What is marker Interface in java?</u>

   i. An empty interface is known as marker interface in java.
   ii. It provides runtime type information about the implemented class objects to the compiler and jvm so that they have some additional information about the objects and it is also known as tagging interface.

8. <u>What is a constructor in java and How many types of constructors are there in java?</u>

   i. Constructors are special type of methods whose name is same as class name.
   ii. Whenever we create an object a constructor is being called.
   iii. Constructors are permanently void so they don't support the return value.

iv. But we can write return in a constructor which will return the cursor to the caller.
v. There are 3 type of constructor present in java.

- **Parameterized:**

The constructor which is having some parameter is known as parameterized constructor.

- **Non-Parameterized**

If we create a constructor which don't have any parameter in it is called non parameterized constructor.

- **Default-Constructor**

If we don't create a constructor in a class at runtime compiler will invoked automatically an empty body constructor that is known as default constructor.

9. What is class in java?

i. A class is a blueprint from where objects are created.
ii. It is a logical entity.
iii. A class in java contain methods, variables, constructors, blocks etc.

10. What is an object in java?

i. In java objects are the blue print of class.
ii. An object is a physical entity which is having some property and behavior.

11. What is this keyword in java?

i. This key word is a special reference variable in java.
ii. It will always points to the current object.

12. What is super keyword in java?

i. Super keyword in java is a special reference variable.
ii. It always points to the immediate parent class object.