Java OOPs interview - Surrounding Questions and Answers

Constructors:

1. How will you call parameter less constructor, write one example?

   Ans: // Output is Default Constructor

```
public class AjayCls {

        public static void main(String[] args) {
                Student s = new Student();
        }
}
class Student{
        public Student(){
System.out.println("This is default Constuctor");
        }
}
```

2. How will you call parameterized constructor, write one example?

   Ans: // Output is Ajay 12

```
public class AjayCls {

        public static void main(String[] args) {
                Student s = new Student("Ajay",12);
                System.out.println(s.sname);
                System.out.println(s.sid);
        }
}
class Student{
        String sname;
        int sid;
        public Student(String sname, int sid){
                this.sid =sid;
                this.sname = sname;
        }
}
```

3. Can we have private constructor?

   Ans: YES, if in case it is made private then it would enforce singleton design pattern.

4. Can I override constructor?

   Ans: NO, Because Constructor don’t have return type.

5. Can I overload constructor?
   Ans: YES, technically we can call as Constructor Overloading.

6. Why we can't override constructor?
   Ans: Because Constructor don't have return type that's why we can't Override.

7. How will you call parent class constructor, which doesn't have parameters?
   Ans: A no-argument constructor(default) is one that doesn't have any parameters, for example public Person(). If a subclass has no call to a superclass constructor using super as the first line in a subclass constructor then the compiler will automatically add a super() call as the first line in a constructor.

8. How will you call parent class constructor, which have parameters?
   Ans: To explicitly call the superclass constructor from the subclass constructor, we use super() . It's a special form of the super keyword. super() can be used only inside the subclass constructor and must be the first statement.

<u>Encapsulation:</u>

1. How will you achieve encapsulation in java?
   Ans: Binding logically related data and functions in a common related place.
   or prevent direct access by making variables private and provide controlled access by making public setter and getter methods.

2. Give me one scenario example where you have implemented encapsulation?
   Ans: If I am writing a hospital application, then I will keep all doctor data and functions in Doctor class, patient data and functions in Patient class one class to achieve encapsulation.

3. What is the use of encapsulation / why do we need encapsulation?
   Ans:
   - **Encapsulation improves code readability:** it is easy to read and understand our project.

- **Encapsulation improves code understandability:** easy to understand the code.
- **Encapsulation improves code maintainability:** we can fix/solve problems easily.

4. How will you hide data from external world?
Ans: prevent direct access by making variables private and provide controlled access by making public setter and getter methods.

5. Can I use encapsulation for hiding data from external world?
Ans: YES

6. Which methods are used to achieve encapsulation?
Ans: setter() and getter()

7. What is bean class [or] what is java bean [or] POJO class (Plain Old Java Object)?
Ans: Binding logically related data and functions in a common related place.
or prevent direct access by making variables private and provide controlled access by making public setter and getter methods.

8. What is setter and getter()?
Ans: **setter() or mutators:** setter is a method that updates the value of a variable.
getter() or accessors: getter is a method that reads the value of a variable.

<u>Inheritance:</u>

1. Can I inherit private variables of parent class in child class?

   Ans: NO
2. What is the use of inheritance?

   Ans: The most important use of inheritance in Java is code reusability (code duplication).
3. Is the constructor inherited to child classes?

   Ans: NO, but the constructor of the parentclass can be invoked from the childclass.

4. Are final methods inherited to child classes?

   Ans: YES, final method is inherited but you cannot override it.

   For Example:

```
class Bike{
	final void run(){
		System.out.println("running...");
	}
}
class Honda2 extends Bike{
	public static void main(String args[]){
		new Honda2().run();
	}
}
```

5. Can I call / access final methods from child class?

   Ans: YES, we can call it but we can’t override.

6. Can I override final methods in the child class?

   Ans: NO

7. How do you call a parent class constructor from a child class?

   Ans: Using super() method.

8. What is object class?

   Ans: The Object class is the parent class of all the classes in java by default. In other words, it is the topmost class of java.

   It has many methods such as,

   1) equals()
   2) hashcode()
   3) getClass()
   4) toString()
   5) clone()
   6) wait()
   7) notify()
   8) notifyall()
   9) finalize()

9. Can I access parent class private variables and private methods, in child class?

   Ans: NO, because we can access private variables and methods in only declared class.

10. If I don't want to inherit my class, what should I do?

    Ans: make the method in super-class private.

11. Can I inherit the final class?

    Ans: NO

12. What is the difference between final class and private class?

    Ans: <u>final class:</u> we can create objects for final classes.

    <u>private class:</u> we can't create object, we can't access parent class variables and methods in child class.

13. Why java doesn't support multiple inheritance?

    Ans: function ambiguity. Example shown below question

14. Show one example for function ambiguity.

    Ans:

```
public class A{
  public void f1(){
          syso("Hi");
  }
}
public class B{
  public void f1(){
          syso("Hello");
  }
}
public class C extends A,B{
    public void f2(){
          f1();
   }
}
```

15. How will you achieve multiple inheritance in java? Show one example?

    Ans: Multiple Inheritance is not possible in java but, we can achieve by using Interface.

16. How many types of inheritances are there in java?

    Ans: There are 5 types of Inheritance in java

    1) Single level Inheritance
    2) Multi-level Inheritance
    3) Hierarchical Inheritance
    4) Multiple Inheritance // its not supported in java
    5) Hybrid Inheritance // it's not supported in java

17. Which keyword do you use for inheritance?

    Ans: extends

18. What is the limitation or disadvantage of inheritance in java

    Ans:

    - Doesn't support Multiple Inheritance.
    - if we don't want parent class functions still, we get those functions it increases code size.

Method overloading:

1. Public void f1() {} — public int f1() {return 10;} is it overloading?

   Ans: NO, because in this code functions having same parameters. as per rule overloading does having different parameters.

2. Can I overload constructor?

   Ans: YES, technically called Constructor Overloading.

3. Can I overload static methods?

   Ans: Yes, we can have two or more static methods with the same name, but differences in input parameters.

4. Can I overload and override final methods?

   Ans: No, the Methods that are declared as final cannot be Overridden or hidden.

5. How will you call overloaded methods, example?

   Ans: When Two or more methods having same method name and different parameters in given class then we can say that method is overloaded.

   Ex:
```
public class A{
            public void m1() {
                        System.out.println("Hi");
            }
            public int m1(int x) {
                        return 3;
            }
    }
```

Method overriding:

1. Can I override private methods of parent class in child class?

   Ans: NO

2. Can I override abstract methods in child class?

   Ans: YES

3. Can I override parent class constructor in child class, why?

   Ans: NO, Constructor don’t have return type.

4. If I don't want to override a parent class method in child class, what should I do?

   Ans: make it parent class methods as final or private.

5. Which type of polymorphism you can achieve with method overriding?

   Ans: Using Runtime Polymorphism (Dynamic polymorphism)

6. What is the difference between private method and final method (as we can’t override both methods in child class)?

   Ans: private method only we can access in declared class. once method is declared as final we cant override that method in child class.

7. Can I override parent class static methods?

   Ans: NO

Polymorphism:

1. Give one example for static polymorphism?

   Ans: Method Overloading is an example for Static Polymorphism (Compile Polymorphism).

   Example:

```
public class Base{
        public void m1() {
                System.out.println("method one");
        }
        public int m1(int x) {
                return 10;
        }
}
```

2. Give one example for dynamic polymorphism?

   Ans: Method Overriding is example for dynamic polymorphism.

```
public class Base{
        public void m1() {
                System.out.println("method one");
        }
        public void m1() {
                System.out.println("method two but same method name");
        }
}
```

3. Show code for achieving dynamic polymorphism?

   Ans:

```
public class AjayCls {

        public static void main(String[] args) {
                Client c = new Client();
                Server s = new Server();
                s.run(c);
        }
}
class Server{
        public void run(Client c) {
                System.out.println("Hello Server");
                System.out.println("server is running.");
                c.done();
        }
}
class Client{
        public void done() {
                System.out.println("Hi");
        }
}
```

<u>NOTE:</u>

<u>UPCASTING:</u>

- Assigning Child class object to Parent class variable is called Upcasting.
- It is used in Dynamic Polymorphism.
- We can access all parent class methods which are not overridden.
- we can access child overridden methods.

Example:

```
public class AjayCls {

            public static void main(String[] args) {
                    A a1 = new B();
                    a1.f1();
                    a1.f2();

            }
    }
```

```
class A{
        public void f1() {
                System.out.println("Hi");
        }
        public void f2() {
                System.out.println("Hello");
        }
}
class B extends A{
        public void f2() {
                System.out.println("Java");
        }
        public void f3() {
                System.out.println("J2ee");
        }
}
```

<u>Static and non-static:</u>

1. How will you access static variables of a class? Show one example?

   Ans: By Using Class Name

   Example:
   ```
   public class AjayCls {
           public static void main(String[] args) {
                   System.out.println(A.y);
           }
   }
   class A{
           public static int y =20;
   }
   ```

2. How will you access non static variables of a class? Show one example?

   Ans: Non static variables nothing but Instance variables so we can create object and access those variables.

   Example:
   ```
   public class AjayCls {
   public static void main(String[] args) {
           A a1 = new A();
           System.out.println(a1.x);
   }
   class A{
           public int x =10;
   }
   ```

3. Which is faster, static methods or instance methods. Why?

   Ans: Static methods are faster when we call using Class name. because we can directly call using class name and control goes directly to code segment and data.

   <u>NOTE:</u>

   - Static variables reduce memory wastage
   - Static variables are faster
   - Static methods shared between all objects.
   - Static variables execute only once in runtime. but static methods execute Nth time.

   note: we can't use super and this inside static methods.

4. Why is the main() method static?

   Ans: JVM code is written in such way that JVM want to call main() method directly without creating object and execute faster.

5. Can I access non-static variables inside a static method?

   Ans: Directly is not possible but indirectly is possible by creating object.

Example:
```
public class AjayCls {
        public static void main(String[] args) {
                A a = new A();
                a.m1();
        }
}
class A{
        public int x =10;
        public static int y =20;

        public static void m1() {
                System.out.println(y);
                System.out.println(x);// In this line error, directly not possible
                A a = new A();// by creating object indirectly possible
                System.out.println(a.x);
        }
}
```

6. What is the difference between static methods and instance methods?

   Ans:

static methods	instance methods
➢ Static methods must be called by using class name ➢ we can't use super and this keywords in static methods	➢ instance methods must be called by using object name. ➢ we can use super and this in instance methods

<u>Access specifier:</u>

1. Can I access protected variables in other packages?

   Ans: YES

2. Can I access private variables in other classes?

   Ans: NO

3. Which access specifier is used for data security?

   Ans: Private

4. Difference between protected and default?

   Ans:

Protected	default
➢ The protected access modifier is accessible within package and outside the package but through inheritance only. ➢ It provides more accessibility than the default modifier	➢ The default modifier is accessible only within package . ➢ It provides more accessibility than private. ➢ it is more restrictive than protected, and public.

Abstract methods and abstract classes:

1. If I have 0 abstract methods in a class, can I make that class an abstract class?

   Ans: YES, Abstract class is class which contains zero or more abstract methods.

2. Can I create an object for an abstract class?

   Ans: NO

3. How will you access methods of an abstract class?

   Ans: Inherited parent class in child class then create object for child class .

4. When will you use an abstract class, give one scenario with code? (Same question can be asked as what is the use of abstract class).

   Ans: An abstract class is used if you want to provide a common, implemented functionality among all the implementations of the component..

   Example:

   You are designing a Bank class. every bank has simpleInterest() functionality to calculate simple interest. But interest rates differ based on the Bank.

   ```
   abstract class Bank{
           public abstract void simpleIntrest();

   }
   ```

5. When will you use an abstract method, give one scenario with code (same question can be asked as what is the use of abstract method)?

   Ans: abstraction means hiding the irrelevant details from the user to focus only on the essential details to increase efficiency thereby reducing complexity.

6. Can I achieve complete or full abstraction with the help of abstract class? Why if not?

   Ans: We can achieve only Partial abstraction.

7. Can I make an abstract class as a final class?

   Ans: NO, final class can not be inherited but abstract class can be inherited.

8. Give one real time example (not java) of abstract class.?

   Ans: You are writing a class for Kid. kid has few functionalities. first functionality is walk(), second functionality is talk(), third functionality is think(), fourth functionality is eat(). How will you design this class?

   every kid eats, but may not think talk and walk. it is incomplete class. so we have to make it as an abstract class.

Abstraction:

1. How will you achieve abstraction in java - which concept will you use?

   Ans: Hiding implementation details and exposing required details to the user

2. Give me one real time example (not java) of abstraction?

   Ans: ATM

3. Do you use interface or abstract class to achieve abstraction, why?

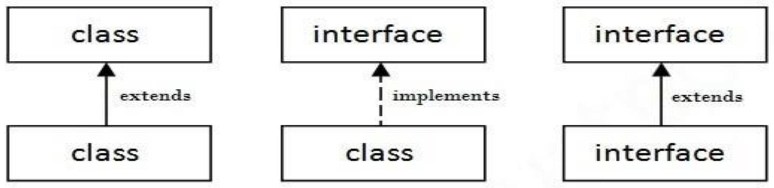
   Ans: Yes

4. Does abstraction provide data security or code security?

   Ans: code security

Interface:

Note:



1. What is the use of interface (same question can be asked as when will you use interface) give one scenario example with code?

   Ans: I want to share common functionalities to un related classes, then I should use interface.

   and we can achieve 100% abstraction.

   interfaces can't have functions with bodies. interfaces can have only method declarations. hence, we can achieve 100% abstraction with interfaces

   Example:

```
public class AjayCls {
	public static void main(String[] args) {
		C c = new C();
		c.f1();
		c.f2();
	}
}

public interface A{
	public void f1();
}
public interface B extends A{
	public void f2();
}
public class C implements B{
	public void f1() {
		System.out.println("Hii");
	}
	public void f2() {
		System.out.println("Hello");
	}
}
```

2. Does interface support multiple inheritance, show code?

   Ans: YES

   Example:

```
public class AjayCls {
	public static void main(String[] args) {
		C c = new C();
		c.f1();
		c.f2();
	}
}
public interface A{
	public void f1();
}
public interface B{
	public void f2();
}
public class C implements A , B{
	public void f1() {
		System.out.println("HI");
	}
	public void f2() {
		System.out.println("Hello");
	}
}
```

3. How will you achieve multiple inheritance with interfaces?

   Ans:

   I don’t know

4. How will you use interface?

   Ans: by using implements

5. What is the access specifier used for the members of interfaces?

   Ans: public

6. Can we achieve complete or full abstraction with an interface? Why

   Ans: yes. Interface does contain only abstract methods.

7. Can I implement multiple interfaces in a class?

   Ans: Yes

8. Give one real world example (not java) of interface?

   Ans: The Interface is a medium to interact between user and system devices. For example, in our real life, if we consider the interface to be given as an example is the air condition, bike, ATM machine, etc.,

   TV --- remote---Person (remote is interface)

<u>Exceptions:</u>

1. What is checked exception?

   Ans: It is mandatory to handle the exception else compiler will throw error.

2. What is unchecked exception?

   Ans: It is not mandatory to handle the exception complier will not throw error.

3. Give one example for checked and unchecked exception?

   Ans: for checked exception:

   IOException, FileNotFoundException

   for unchecked exception:

   ArrayIndexOutOfBoundsException, ArithmeticException

   NullPointerException, ClassCastException

4. If I don't want to handle an exception, what should I do?

   Ans: Adding a throws clause to the method declaration.

5. If I use throws, and if an exception occurs in the current method, then what will happen. Who is going to handle that exception?

   Ans: When a method declares that it throws an exception, it is not required to handle the exception. The caller of a method that throws exceptions is required to handle the exceptions (or throw them to its caller and so on) so that the flow of the program can be maintained.

6. What is the use of throw?

   Ans:

   - The throw keyword is used to create a custom error.
   - After catching the exception if we want to rethrow an exception we use throw keyword.
   - used to raise (throw) the custom exception.

7. How will you create custom exception?

   Ans:
```
public class WrongFileNameException extends Exception {
        public WrongFileNameException(String errorMessage) {
        super(errorMessage);
       }
}
```

8. What happens if an exception occurs in the finally block? Will the finally block be executed?

   Ans: NO

9. If we use System.exit(), then finally block will be executed or not?

   Ans: NO

<u>Collections & Generics:</u>

1. What is the use of Collections?

   Ans: Java Collections are the one-stop solutions for all the data manipulation jobs such as

   - storing data,
   - searching,
   - sorting,
   - insertion,
   - deletion, and
   - updating of data.
   - Java collection responds as a single object, and
   - a Java Collection Framework provides various Interfaces and Classes.

2. How will you store 10, “ramesh”, 10.5 without using class and without using 3 variables?

   Ans:
```
Object[ ] arr = {10, “ramesh”, 10.5};
    OR
  Collection c =new Collecion():
    c.add(10);
    c.add(“ramesh”);
    c.add(10.5);
System.out.println(c);
```

3. What is the difference between Object[] array and collections?

   Ans:

4. In which version java collections are introduced?

   Ans: Version 1.2

5. What is the difference between collections and generics?

   Ans:

Collections	Generics

• Collections are note safe to use • While using Collections we don't have to specify the datatype of elements.	• Generics are safe to use • While using Generics we have to specify the datatype of elements.

6. Why generics are safe?
   Ans: Generics were added to Java to ensure type safety.

7. Which one is faster, collections or generics?
   Ans: Generics is faster.

8. Difference between linked list and array list which is best
   Ans: LinkedList is faster being node based as not much bit shifting required.

ArrayList	LinkedList
• Arraylist elements are stored in sequential order • ArrayList elements are faster compared to LinkedList	• LinkedList elements are stored in random order. • LinkedList are slow(When adding elements in beginning and deleting elements in ending and modify and insertion the elements that time LinkedList is faster)

9. When will you use linked list (in what cases linked is faster)?
   Ans: When adding elements in beginning and deleting elements in ending and modify and insertion the elements that time LinkedList is faster.

10. If linked list is slow why should I use?
    Ans: LinkedList internally uses a doubly linked list to store the elements. Its stores the elements with nodes.

11. difference between list and set [or] arraylist vs set?

Ans:

List	Set
➢ List stores the elements in insertion order. ➢ List allow duplicate elements.	➢ Set stores elements in the random order. ➢ Set doesn't allow duplicate elements

ArrayList	Set
➢ ArrayList stores the elements in sequential order. ➢ ArrayList allows duplicate values	➢ Set stores elements in the random order. ➢ Set doesn't allow duplicate elements

12. Difference between Set and Map?

Ans:

Set	Map
➢ Set stores elements in the random order. ➢ Set doesn't allow duplicate elements	➢ Map is used to stores pair of values. ➢ Key should be unique and value can be duplicate.

13. Difference between Array and List?

Ans:

S.No.	List	Array
2	List cannot manage arithmetic operations.	Array can manage arithmetic operations.
3	It consists of elements that belong to the different data types.	It consists of elements that belong to the same data type.

4	When it comes to flexibility, the list is perfect as it allows easy modification of data.	When it comes to flexibility, the array is not suitable as it does not allow easy modification of data.
5	It consumes a larger memory.	It consumes less memory than a list.
6	In a list, the complete list can be accessed without any specific looping.	In an array, a loop is mandatory to access the components of the array.
7	It favors a shorter sequence of data.	It favors a longer sequence of data.