Java Interview Questions – Part I

Softech Solutions Inc.

www.softechsolutionsgroup.com

saney.alam@softechsolutionsgroup.com

Java Interview Questions – Part I

Q1: What is the difference between Encapsulation and Abstraction?

Ans.

1. Abstraction solves the problem at design level while encapsulation solves the problem at implementation level.
2. Abstraction is used for hiding the unwanted data and giving relevant data. While Encapsulation means hiding the code and data into a single unit to protect the data from outside world.
3. Abstraction lets you focus on what the object does instead of how it does it while Encapsulation means hiding the internal details or mechanics of how an object does something.
4. For example: Outer Look of a Television, like it has a display screen and channel buttons to change channel it explains Abstraction but Inner Implementation detail of a Television how CRT and Display Screen are connect with each other using different circuits , it explains Encapsulation.

Q2: Can you give a real world example of Encapsulation and Abstraction?

Ans. Car Engine is an example of encapsulation and abstraction. You ignite the car using an interface called starter and least bothered about how the tire actually moves (This is abstraction). The engine encapsulates the complete process to itself only and doesn't allow you to start the other components like the radiator etc. (this is encapsulation).

Q3: Differences between abstract class and interface?

Ans. Abstract classes can have both abstract methods (method declarations) as well as concrete methods (inherited to the derived classes) whereas Interfaces can only have abstract methods (method declarations).

Q4: Difference between Abstract and Concrete Class?

Ans. Abstract classes are only meant to be sub classed and not meant to be instantiated whereas concrete classes are meant to be instantiated.

Q5: Difference between Factory and Abstract Factory Design Pattern?

Ans. Factory Pattern deals with creation of objects delegated to a separate factory class whereas Abstract Factory patterns works around a super-factory which creates other factories.

Q6: What are the design considerations while making a choice between using interface and abstract class?

Ans. Keep it as a Abstract Class if its a "Is a" Relationsship and should do subset/all of the functionality. Keep it as Interface if its a "Should Do" relationship.

Q7: How is Abstraction implemented in Java?

Ans. Abstraction is provided in Java by following ways

Coding to the (Interfaces / Abstract Classes) or contracts

By encapsulating details within classes and exposing the minimal Door (few public methods).

Q8: If an Abstract class has only abstract methods, What's the difference between such a class and an interface?

Ans. Such a class still can have member elements which can be inherited and hence facilitate code reuse. Moreover Abstract class can have non final static elements whereas interfaces are only allowed to have static final elements.

Q9: What is an abstract class?

Ans. Abstract class is the class that is not supposed to be instantiated. The purpose of the class to only have extension to the derived class.

Q10: Can we declare an abstract method private ?

Ans. No Abstract methods can only be declared protected or public.

Q11: Difference between Abstraction and Implementation hiding ?

Ans. Implementation Hiding is a broader concept. Abstraction is a way of implementation hiding in OOP's.

Q12: What are the examples of Abstraction in Java ?

Ans. function calling - hides implementation details wrapper classes new operator - Creates object in memory, calls constructor.

Q13: what will be the output ?

```
class Animal {
    public void eat() throws Exception {
    }
}
class Dog2 extends Animal {
    public void eat(){}
  public static void main(){
```

```
    Animal an = new Dog2();
    an.eat();
  }
}
```

Ans. Compile Time Error: Unhandled exception type Exception.

Q14: Will this code Work ? If not , Why ?
java.util.Calendar c = new java.util.Calendar();

Ans. No. It gives the error "Cannot Instantiate the type Calendar". Calendar is an abstract class and hence Calendar object should be instantiated using Calendar.getInstance().

Q15: Is java.util.Date an abstract Class ? Is java.util.Calendar an abstract Class ?
Ans. Date is not a abstract class whereas Calendar is.

Q16: Will this code compile ?

```
  public class BuggyBread1{
        abstract public void test();
  }
```

Ans. No. It will give the compilation error saying "The abstract method test in type BuggyBread1 can only be defined by an abstract class". We need to declare the class abstract for it to have any abstract method.

Q17. Will this Code compile ?

```
abstract public class BuggyBread1{

      abstract public void test(){};

}
```

Ans. No. This will give a compilation error saying "Abstract methods do not specify a body".

Q18: What is ADT or Abtstract Data Type ?

Ans. ADT is a container which holds different types of objects with specifications.

For example - Stack, Array, Liked list, Tree.

Q19: Why can't we declare a class abstract as well as final ?

Ans. Abstract means that the class is only meant to be subclasses whereas final means that it cannot be subclasses so both concepts - abstract and final are actually mutually exclusive and hence not permitted together.

Q20: In the following code , how many methods needs to be implemented in Class B ?

```
public interface A{

        public void method1();

        public void method2();

        public void method3();

}

abstract class B implements A{

}
```

Ans. As Class B has been declared abstract , we can either implement any of these methods and just declare rest of them abstract.

Q21: What should a class do if its extending an abstract class ?

Ans. It should either implement the abstract methods or re-declare them abstract.

Q22: Can we declare a main method as abstract ?

Ans. No. Static methods cannot be overridden and hence make no sense to be declared abstract.

Q23: When should we use Abstract classes ?

Ans. Abstract classes provide a mechanism of interfacing (using abstract method) as well as inheritance (extending abstract class). So they should be used in place of interfaces in case there is some code (methods) or object body (member elements) that can be reused with inheritance.

Q24: Can we overload abstract methods ?

Ans. Yes

Q25: Can we override abstract methods ?

Ans. Abstract methods are the methods that need to be overridden in the derived class (either as implementing method or again abstract method) so it's not only allowed but its required to override abstract method in the derived class.

Q26: Can we override an abstract method with abstract method in the derived class ?

Ans. Yes, but in that case the derived class itself should be abstract. We cannot have an object of a class without definition for the method which has been declared abstract in the parent.

Q27: What was the need of creating an abstract class in Java when we cannot instantiate an object of it ?

Ans. The objective of an abstract class is to provide an interface as well as code reuse. Though we cannot instantiate an instance of abstract class but we can reuse the code by extending such a class. Moreover abstract class provides interfacing to outside components just like interfaces.

Q28: Why can't we create an instance of abstract class ? Why is it restricted by compiler ?

Ans. Because Abstract class is incomplete i.e there are abstract methods that needs to be defined by extending classes.

Q29: What is the importance of Abstract Class ?

Ans. Abstract classes provide a mechanism of interfacing (using abstract method) as well as code reuse through inheritance (extending abstract class)

Comparing to concrete class they have an advantage of providing interface which a concrete class doesn't provide.

Comparing to interfaces they have an advantage of providing code reuse through inheritance which interfaces don't provide.

Q30. Which of the following can be declared abstract?

Ans. instance methods