1. Give an example among tight coupling and loose coupling.

**Ans :**

1. Write in brief abstract class.

**Ans :**

● Abstract class is a class which has been declared abstract by using the ‘abstract’ keyword.

● Ex.

abstract class Test {

//Code

}

● An abstract class may or may not have any abstract method.

● But if any class has any abstract method then class must be declared as abstract.

● We can not create object of an abstract class.

● Abstract class data and methods can be used through child class' objects only.

● It can have [constructors](https://www.javatpoint.com/java-constructor) and static methods also.

● It can have final methods which will force the subclass not to change the body of the method.

//Example of an abstract class that has abstract and non-abstract methods.

 abstract class Bike

{

   Bike()

{

System.out.println("bike is created");

}

   abstract void run();

   void changeGear()

{

System.out.println("gear changed");

}

 }

//Creating a Child class which inherits Abstract class

 class Honda extends Bike

{

  void run()

{

System.out.println("running safely..");

}

 }

//Creating a Test class which calls abstract and non-abstract methods

 class TestAbstraction2

{

 public static void main(String args[])

{

   Bike obj = new Honda();

   obj.run();

   obj.changeGear();

 }

}

1. Define the Benefits of oops over pop?

**Ans :**

● Encapsulation is used to hide the data.

● The existing code can be reused.

● Adding new data and functions is easy.

● Inheritance property is used.

● Used for solving big problems.

1. Explain Generalization and Specialization?

**Ans :**

● Generalization : The process of converting subclass type into superclass type is called generalization in java. This is because we are making the subclass to become more general so that its scope can be more widening.

This conversion is also called widening or upcasting in referenced data types.

● Specialization : The conversion of a superclass type into subclass type is called specialization in java.

Specialization means going down from a more general form to a more specific form. Thus, its scope will be narrowed. Hence, this conversion is also called narrowing or down-casting in referenced data types.

Specialization is not safe because classes will be more and more specific. In this case, we will need cast operator.

Ex. Let us consider a superclass Mobile and subclasse Samsung whose subclass is Samsung Galaxy.

When we talk about a mobile, In general, it may represent any kind of mobile. So, here, the scope is widened.

Now, suppose we talk about Samsung mobile, then we come down one step in the hierarchy of inheritance and we have eliminated any other kind of mobiles.  
Thus, we are becoming more specific. When we still come down to Samsung Galaxy, we are pointing only Samsung Galaxy mobile and not any other Samsung mobile.

Thus, this is very specific. This means that when we move down from superclass to subclasses, we are becoming more and more specific.

1. Write in brief Association, Aggregation and Composition?

**Ans :**

● Association :

Association relationship is a structural relationship in which different objects are linked within the system. It exhibits a binary relationship between the objects representing an activity. It depicts the relationship between objects, such as a teacher, can be associated with multiple teachers.

It is represented by a line between the classes followed by an arrow that navigates the direction, and when the arrow is on both sides, it is then called a bidirectional association. We can specify the multiplicity of an association by adding the adornments on the line that will denote the association.

The composition and aggregation are two subsets of association.

● Aggregation :

Aggregation is a subset of association, is a collection of different things. It represents has a relationship. It is more specific than an association. It describes a part-whole or part-of relationship. It is a binary association, i.e., it only involves two classes. It is a kind of relationship in which the child is independent of its parent.

For example:Here we are considering a car and a wheel example. A car cannot move without a wheel. But the wheel can be independently used with the bike, scooter, cycle, or any other vehicle. The wheel object can exist without the car object, which proves to be an aggregation relationship.

● Composition :

The composition is a part of aggregation, and it portrays the whole-part relationship. It depicts dependency between a composite (parent) and its parts (children), which means that if the composite is discarded, so will its parts get deleted. It exists between similar objects.

Example: If a file is placed in a folder and that is folder is deleted. The file residing inside that folder will also get deleted at the time of folder deletion.

1. Write in brief Object Composition vs. Inheritance.

**Ans :**

| S.NO | Inheritance | Composition |
| --- | --- | --- |
| 1. | In inheritance, we define the class which we are inheriting(super class) and most importantly it cannot be changed at runtime | Whereas in composition we only define a type which we want to use and which can hold its different implementation also it can change at runtime. Hence, Composition is much more flexible than Inheritance. |
| 2. | Here we can only extend one class, in other words more than one class can’t be extended as java do not support multiple inheritance. | Whereas composition allows to use functionality from different class. |
| 3. | In inheritance we need parent class in order to test child class. | Composition allows to test the implementation of the classes we are using independent of parent or child class. |
| 4. | Inheritance cannot extend final class. | Whereas composition allows code reuse even from final classes. |
| 5. | It is an is-a relationship. | While it is a has-a relationship |

1. Explain cohesion?

**Ans :**

● Cohesion in Java is the Object-Oriented principle most closely associated with making sure that a class is designed with a single, well-focused purpose. In object-oriented design, cohesion refers to how a single class is designed.

●  The more focused a class is, the more is the cohesiveness of that class.

● The advantage of high cohesion is that such classes are much easier to maintain (and less frequently changed) than classes with low cohesion. Another benefit of high cohesion is that classes with a well-focused purpose tend to be more reusable than other classes.

● Example: Suppose we have a class that multiplies two numbers, but the same class creates a pop-up window displaying the result. This is an example of a low cohesive class because the window and the multiplication operation don’t have much in common. To make it high cohesive, we would have to create a class Display and a class Multiply. The Display will call Multiply’s method to get the result and display it. This way to develop a high cohesive solution.

1. Explain “black-box-reuse” and “white-box-reuse”?

**Ans :**

1. Explain “this”

**Ans :**

● Reference to the current object. That is, this is always a reference to an object on which method was invoked.

● ‘this’ can be used to call the overloaded constructor from the other constructors within the same class. But it should always be the first statement within the constructor.

Ex. Cricketer(int totRun)

{

this(); //Calling no argument constructor of Cricketer class.

}

● this ( ) can not be used within any other methods other than constructors

Ex. void updateTotalRuns(int totRuns)

{

this(); //Compilation Error.

}

1. Write in brief static member and member functions.

**Ans :**

Static Members :

Variables and methods declared using keyword static are called static members of a class. As we know that non-static variables and methods belong to instance. But static members (variables, methods) belong to class. Static members are not part of any instance of the class. Static members can be accessed using class name directly, in other words, there is no need to create instance of the class specifically to use them.  
Static members can be of two types:  
-- Static Variables :Static variables are also called class variables because they can be accessed using class name, whereas, non static variables are called instance variables and can be accessed using instance reference only. Static variables occupy single location in the memory. These can also be accessed using instance reference  
-- Static Methods :Static methods are also called class methods. A static method belongs to class, it can be used with class name directly. It is also accessible using instance references.  
Static methods can use static variables only, whereas non-static methods can use both instance variables and static variables.