**31. Difference among overloading and overriding ?**

|  |  |
| --- | --- |
| **Method overloading**   * Method overloading is a compile-time polymorphism. * It helps to increase the readability of the program * Static binding is being used for overloaded methods. * Poor Performance due to compile time polymorphism. * Argument list should be different while doing method overloading. * Private and final methods can be overloaded. | **Method overriding**     * Method overriding is a run-time polymorphism. * It is used to grant the specific implementation of the method which is already provided by its parent class or superclass. * Dynamic binding is being used for overriding methods. * It gives better performance. The reason behind this is that the binding of overridden methods is being done at runtime. * Argument list should be same in method overriding. * Private and final methods can’t be overridden |

**Method Overloading :**

In Method Overloading, Methods of the same class shares the same name but each method must have a different number of parameters or parameters having different types and order.

Method Overloading is to “add” or “extend” more to the method’s behavior.

It is a compile-time polymorphism.

The methods must have a different signature.

It may or may not need inheritance in Method Overloading.

Let’s take a look at the example below to understand it better.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | class Adder {  Static int add(int a, int b)  {  return a+b;  }  Static double add( double a, double b)  {  return a+b;  }  public static void main(String args[])  {  System.out.println(Adder.add(11,11));  System.out.println(Adder.add(12.3,12.6));  }} |

**Method Overriding:**

In Method Overriding, the subclass has the same method with the same name and exactly the same number and type of parameters and same return type as a superclass.

Method Overriding is to “Change” existing behavior of the method.

It is a run time polymorphism.

The methods must have the same signature.

It always requires inheritance in Method Overriding.

Let’s take a look at the example below to understand it better.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | class Car {  void run(){  System.out.println(“car is running”);  }  Class Audi extends Car{  void run()  {  System.out.prinltn("Audi is running safely with 100km");  }  public static void main( String args[])  {  Car b=new Audi();  b.run();  }  } |

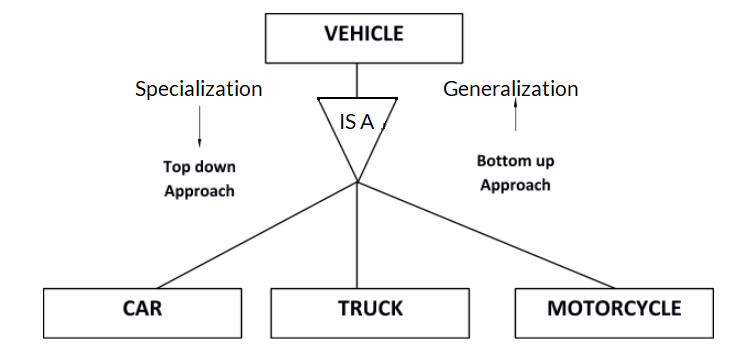
**32. Whether static method can use non-static members?**

A static method is a method that belongs to a class, but it does not belong to an instance of that class and this method can be called without the instance or object of that class. Every method in java defaults to a non-static method without static keyword preceding it. Non-static methods can access any static method and static variable, without creating an instance of the object. A static method can only access static data members and static methods of another class or same class but cannot access non-static methods and variables. Also, a static method can rewrite the values of any static data member. A non-static method can access static data members and static methods as well as non-static members and methods of another class or same class, also can change the values of any static data member.

**33. Explain a base class, sub class, super class?**

**Subclasses /Child class=**

A subclass is a class derived from the superclass. It inherits the properties of the superclass and also contains attributes of its own. An example is:



Car, Truck and Motorcycle are all subclasses of the superclass Vehicle. They all inherit common attributes from vehicle such as speed, colour etc. while they have different attributes also i.e Number of wheels in Car is 4 while in Motorcycle is 2.

**Superclasses /Parent class /Base class=**

A superclass is the class from which many subclasses can be created. The subclasses inherit the characteristics of a superclass. The superclass is also known as the parent class or base class.

In the above example, Vehicle is the Superclass and its subclasses are Car, Truck and Motorcycle.

**Inheritance=**

Inheritance is basically the process of basing a class on another class i.e to build a class on a existing  class. The new class contains all the features and functionalities of the old class in addition to its own.

The class which is newly created is known as the subclass or child class and the original class is the parent class or the superclass.

**34. Write in brief linking of base class, sub class and base object, sub object.**

Object relation of Superclass (parent) to Subclass (child) exists while child to parent object relation never exists. This means that reference of parent class could hold the child object while child reference could not hold the parent object.

In case of overriding of non static method the runtime object would evaluate that which method would be executed of subclass or of superclass. While execution of static method depends on the type of reference that object holds.

Other basic rule of inheritance is related to static and non static method overriding that static method in java could not be overridden while non static method can be. However subclass can define static method of same static method signature as superclass have hut that would not be consider as overriding while known as hiding of static method of superclass.

So difference between referencing using superclass reference and referencing using subclass reference is use superclass referencing can holds object of subclass and could only access the methods which are defined/overridden by subclass while use subclass referencing can not hold object of superclass and could access the methods of both superclass and subclass.

**35. Explain an interface?**

Inheritance in Java is the concept where the properties of one class can be inherited by the other. It helps to reuse the code and establish a relationship between different classes. Inheritance is performed between two types of classes:

Parent class (Super or Base class)

Child class (Subclass or Derived class)

A class which inherits the properties is known as Child Class whereas a class whose properties are inherited is known as Parent class.

**Types of inheritance in Java**

Java supports four types of inheritance which are:

Single Inheritance: In single inheritance, one class inherits the properties of another i.e there will be only one parent as well as one child class.

Multilevel Inheritance: When a class is derived from a class which is also derived from another class, i.e. a class having more than one parent class but at different levels, such type of inheritance is called Multilevel Inheritance.

Hierarchical Inheritance: When a class has more than one child classes (subclasses) or in other words, more than one child classes have the same parent class, then such kind of inheritance is known as hierarchical.

Hybrid Inheritance: Hybrid inheritance is a combination of two or more types of inheritance.

**36. Explain exception handling?**

Exception Handling in Java is one of the effective means to handle the runtime errors so that the regular flow of the application can be preserved. Java Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.

Exception is an unwanted or unexpected event, which occurs during the execution of a program, i.e. at run time, that disrupts the normal flow of the program’s instructions. Exceptions can be caught and handled by the program. When an exception occurs within a method, it creates an object. This object is called the exception object. It contains information about the exception, such as the name and description of the exception and the state of the program when the exception occurred.

**Major reasons why an exception Occurs**

Invalid user input

Device failure

Loss of network connection

Physical limitations (out of disk memory)

Code errors

Opening an unavailable file

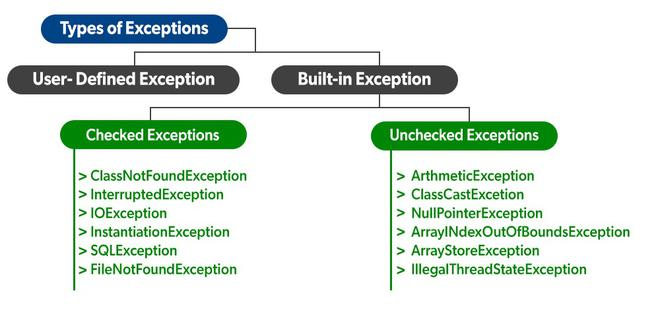
Errors represent irrecoverable conditions such as Java virtual machine (JVM) running out of memory, memory leaks, stack overflow errors, library incompatibility, infinite recursion, etc. Errors are usually beyond the control of the programmer, and we should not try to handle errors.

**Differences between Error and Exception**

Error: An Error indicates a serious problem that a reasonable application should not try to catch.

Exception: Exception indicates conditions that a reasonable application might try to catch.

**Types of Exceptions:**



**37. Explain the difference among structure and a class?**

**Class :**

Class is a blueprint or set of instructions to build specific types of objects. It consists of a list of data members, and a set of operations are performed on the class.

It is a user-defined data type that holds its own data members.

* Class is a basic building block of object-oriented programming (OOP) languages.
* Class is a logical abstraction.
* It stores multiple types of data and is also used to store functions.
* To access the class members, you need to create an instance of the class called objects.

**Syntax :**

class Bike

{

public:

int year;

string model\_name;

string brand\_name;

};

**Structure :**

The Structure is a collection of various types of variables, and these variables can be built-in or user-defined datatypes.

* Structure elements are stored at contiguous memory locations.
* A structure can have static data members, access specifiers, and support inheritance.
* Members of the Structure can be passed to a function
* Members in the Structure can be accessed using the dot (.) operator.

**Syntax :**

struct StructureName

{

member 1;

member 2;

……….

……….

member N;

};

**Differences and similarities between Structure and Class**

* Structure is a value type, while classes are reference types.
* Structure is not inheritable, while classes are.
* Classes use Heap allocation while Structure uses stack allocation.
* Structure must have at least one non-shared variable while class can be empty.
* By default, all structure elements are public, while in-class variables and constants are private, and the others are public.
* Both class and Structure are container type, i.e., they can contain others as a member.
* Both can implement interfaces and have shared constructors, with or without parameters

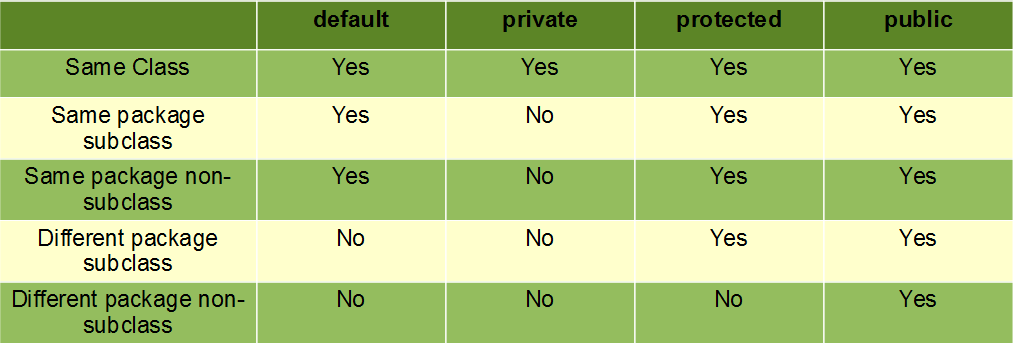
**38. Explain the default access modifier in a class?**

Default access modifier means we do not explicitly declare an access modifier for a class, field, method, etc.

A variable or method declared without any access control modifier is available to any other class in the same package. The fields in an interface are implicitly public static final and the methods in an interface are by default public.

When no access modifier is specified for a class, method, or data member – It is said to be having the default access modifier by default.

The data members, class or methods which are not declared using any access modifiers i.e. having default access modifier are accessible only within the same package.



**39. Explain a pure virtual function?**

Pure virtual functions are also called ‘do nothing functions’.  
Example :

virtual void abc() = 0;

When a pure virtual function is declared in the base class, the compiler necessitates the derived classes to define those functions or redeclare them are pure virtual functions. The classes containing pure virtual functions cannot be used to declare objects of their own. Such classes are called as abstract base classes. A pure virtual function is a function which has no definition in the base class. Its definition lies only in the derived class i.e it is compulsory for the derived class to provide definition of a pure virtual function. Since there is no definition in the base class, these functions can be equated to zero.

Pure virtual function is the function in the base class with no body. Since no body, you have to add the notation =0 for declaration of the pure virtual function in the base class.  
The base class with pure virtual function can't be instantiated since there is no definition of the function in the base class.  
It is necessary for the derived class to override pure virtual function.  
This type of class with one or more pure virtual function is called abstract class which can't be instantiated, it can only be inherited.

**40. Explain dynamic or run time polymorphism?**

Runtime polymorphism or dynamic method dispatch is a process in which a call to an overridden method is resolved at runtime rather than at compile-time. In this process, an overridden method is called through the reference variable of a superclass.

In this process, an overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable

|  |  |
| --- | --- |
| **compile-time polymorphism** | **Runtime polymorphism** |
| * In compile-time polymorphism, call to a method is resolved at compile-time. * It is also known as static binding, early binding, or overloading * Overloading is a way to achieve compile-time polymorphism in which, we can define multiple methods or constructors with different signatures. * It provides fast execution because the type of an object is determined at compile-time * Compile-time polymorphism provides less flexibility because all the things are resolved at compile-time. | * In runtime polymorphism, call to an overridden method is resolved at runtime. * It is also known as dynamic binding, late binding, overriding, or dynamic method dispatch. * Overriding is a way to achieve runtime polymorphism in which, we can redefine some particular method or variable in the derived class. By using overriding, we can give some specific implementation to the base class properties in the derived class. * It provides slower execution as compare to compile-time because the type of an object is determined at run-time. * Run-time polymorphism provides more flexibility because all the things are resolved at runtime. |