**Object class:**

* Object class is a default super most class in java
* If a class does not extend any other class explicitly, then by default that class extends Object class.
* The classes that we create in java are either direct/indirect sub-class to Object class.
* Since object class is the super most class, any type can be upcasted to object type.
* Since upcasting is implicit, we can assign an instance of any type into a reference variable of type Object.

**Polymorphism:**

* An object transforms into many other forms is known as polymorphism.
* In java we can achieve polymorphism through upcasting.
* If a sub-class instance is referred from super class variable, then sub-class instance shows only the features of super class (other features are hidden).
* Note: compiler allows to call only methods which are defined in reference variable type.

**Interface:**

* Interface is also a type definition block in java.
* Interface comes with a contract.
* Interface is 100% abstract
* Interface allows only static and final variables. By default, all variables are static and final
* All the methods defined in interface are by default public and abstract
* Object can’t be created for interface.
* We use implements keyword to implement members from interface
* When a class implements interface, in the sub class, either complete inherited methods or declare the sub-class as abstract otherwise java compiler throws error.
* While overriding interface methods in sub-class, sub-class methods should be public.
* Interface will not have constructors.
* To achieve multiple inheritance, we make use of interface
* A class can implements an interface.
* An interface extends another interface
* A class can implements any interface i.e. class A implements B,C(here B and C are interfaces)-this is multiple inheritance
* A class can extends another class and implement an interface

i.e. class A extends B implements C(first write extends then implements)

* Interface cannot implement a class

**Advantages of Interface:**

1.We can achieve 100% abstraction.

2.We can achieve multiple inheritance.

**Binding:** Connecting method call and method body is called as binding.

There are 2 types of binding;

**1.Early Binding:** In Early binding java compiler connects a method call to a method body at compile time. For static method calls, private method calls and final method calls, java compiler binds method call to method body.

Early binding is also called as Static binding. Method overloading is called as static binding.

**2.Late Binding:** In late binding JRE decides which method body to execute based on the object created at run time. For non-static method calls, non-private method calls and non-final method calls, late binding takes place.

Late binding is also called as dynamic method dispatch. Method overriding is called as Late binding.

**Runtime Polymorphism:**

Runtime polymorphism is a process in which call to an overridden method is resolved at runtime. Because of runtime polymorphism, when an overridden method is called, method body will be executed based on the instance created not based on the reference variable type.

Without inheritance and interface/abstract, we can’t achieve runtime polymorphism.

**Naming Conventions:**

1.Class name should start with upper case letter: public class Class1

2.Method name should start with lowercase letter: public void testMethod()

3.Variable name should start with lowercase letter: int variableName=10;

**Packages:** Package is a collection of classes or interfaces.

package name generally will be in lower case and to create a unique package structure, create a package in the reverse order of your website.

E.g.: [www.abc.com](http://www.abc.com) then package name will be “com.abc”

To refer a class of different package, either use fully classified class name or import a package using import keyword.

Import packageName.\* or import packageName.All :- To import all the classes from particular package

Built-in packages: We get these packages when we install JDK

**1.java.lang** package**:** we have Object class, System class, String class in this package

**2.java.util** package**:** we have ArrayList class ,HashMap class

**3.java.io** package**:** File class, FileWritter class

Java.lang package is a default package. All classes of this package are implicitly imported.

**Access Specifiers:** It sets the visibility

We have 4 types of visibility:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Inside a class | Inside a package(different classes) | Outside a package |
| **Private** members can be accessed | Yes | No | No |
| **Default** members can be accessed | Yes | Yes | No |
| **Protected** members can be accessed | Yes | Yes | Yes (through IS-A relationship) |
| **Public** members can be accessed | Yes | Yes | Yes |

**Encapsulation:** It’s the processes of wrapping the code and data together into a single unit.

* It’s a technique of making fields private and providing access to private fields through public methods.
* Since we are making fields private, encapsulation is also called as data hiding.
* Public methods which are used to SET/GET private variable values are known as setters/getters.

**Advantages:**

1. A class can have total control over the data which is stored in a field.
2. A class can change the implementation without affecting the user’s program.
3. A class can make field read only or write only.
4. Users no need to worry how the data is stored.