Object Oriented Programming

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation.

Object -

- have some state & behavior
- can be defined as an instance of class
- contains an address & takes up some space in memory.

Eg → a white board, ballpen, a acer laptop etc.

Class -

- is a blueprint for the object.
- is a logical entity.
- doesn't consume space.
- objects can be created by class.
- Ex - board class, pen class, laptop class.

Inheritance -
is a method which acquires all the properties and behaviors of a parent object. It represents a IS-A relnship known as parent-child relnship.

Polymorphism -
is a way of performing a single operation in different ways. Polymorphism means many forms.

Abstraction -
is a process of hiding the implementation details and showing only functionality of the program to the user.

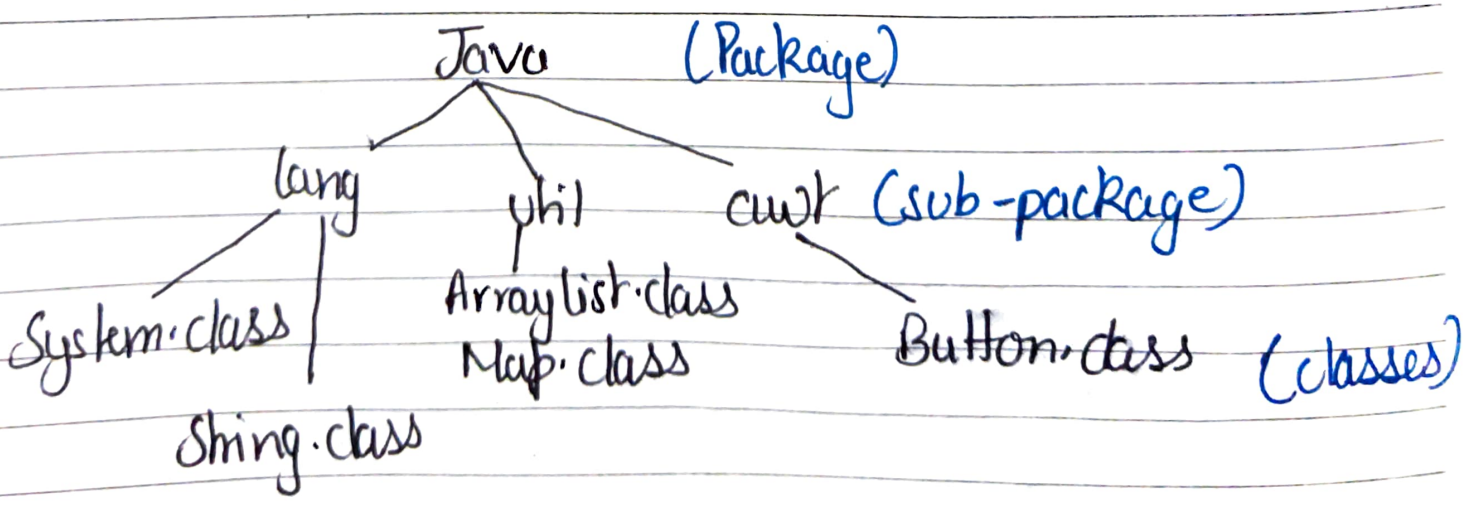
Encapsulation -
is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit.

Encapsulation

Encapsulation in java is a process of wrapping code and data together into a single unit. The data in a class is hidden from other classes, so it is also known as data hiding.

Java Packages
Access Specifier

* Java Packages - is a group of similar types of classes, interfaces and sub-packages. There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql, etc.



Advantages of Packages-

1) used to categorize the classes & interfaces so that they can be easily maintained.
2) provides access protection
3) removes naming collision.

* Access Modifier- in java specifies the accessibility or scope of a field, method, constructor, or class. We can change the access level of fields, constructors, and class by applying the access modifier on it.

Four types of access modifier-

i) Private- access level within the class, can not be accessed outside of class.

ii) Default - access level within the package. If access level is not specified, it will be default.

iii) Public- access level everywhere, can be access within and outside of both classes and packages.

iv) Protected - access level within package & outside package through child class.

Why Encapsulation -

- Keeps related fields & methods together, makes code cleaner & easy to read.
- Controls modification of data fields.

Achieve Encapsulation -

i) declare variables as private.

ii) Provide getter & setter. Done.

Encapsulation refers to the bundling of related fields & methods together. This allows us to achieve data hiding. Encapsulation in itself is not data hiding.

Inheritance -

* Inheritance is an IS-A relationship. We use inheritance only if an is-a relationship is present between the two classes.

Example - ① A car is a vehicle.
② A dog is a animal.

* protected is assigned over methods and fields. Protected members are accessible -
  - From within the class
  - within its subclass
  - within the same package

* Object class of Java is by-default parent class of each classes created

* If their is a common function both in parent & children class than children class person is executed; parent's function is overrided.

* instanceof used to get the parent class object, return type boolean

* Suppose Animal is a parent class, Cat is child class.

Upcasting-

```
Cat c = new Cat();
Animal a = c;
```

Down casting-

```
Cat c = new Cat();
Animal a = c;
Cat cl = (cat)a;
```

* Super Keyword -

In java, super is used to refer to immediate parent class of a child class. super is used by subclass whenever it need to refer to its immediate super class

* The problem with multiple inheritance is that two classes may define different ways of doing the same thing & the subclass can not choose which one to pick, also introduces the Diamond Problem.

* Singleton Pattern is mostly used in multi-threaded & database applications. It is used in logging, caching, thread pools, configuration settings etc.

- Private constructor to restrict instantiation of the class from other classes.
- Private static variables of the same class that is the only instance of the class.
- Public static methods that returns the instance of the class, this is the global access point for outer world to get instance of the singleton class

Polymorphism-

Runtime Polymorphism	Compile-Time Poly.
• also known as dynamic binding. Late binding & overriding as well.	• Static binding. Early binding & overloading as well.
• Overriding is run-time polymorphism having same method with same parameters or signature, but associated in a class & its subclass.	• Overloading is compile time polymorphism where more than one methods share same name with diff parameter/signature & diff return type.
• Slower execution	• Faster Execution

Abstraction

- Data abstraction is the process of hiding certain details & showing only essential information to the user.
- It helps in reduce programming complexity & effort.
- Abstraction can be achieved with either abstract classes or interfaces

Abstract Class-

- Keyword abstract is used before class keyword
- Class object can not be created.
- It contains abstract method; which don't have method body.
- It can contain non-abstract method known as concrete method, having proper method body.
- Abstract class must be extended & abstract method must be overridden

Achieved by- abstract keyword
interface

* Final Keyword -

final keyword can be used for variables, classes and methods.

- final variable can not be changed
- final method can not be overridden
- final class can not be extended

final variables -

- initialized only once.
- must be initialize else compile-time error.
- name final variable in Upper case letter.
- called blank final variable, if it is not initialized while declaration.
- In case of a reference final variable, internal state of the object pointed by that reference variable can be changed.
- Local variable can be final, if not initialized