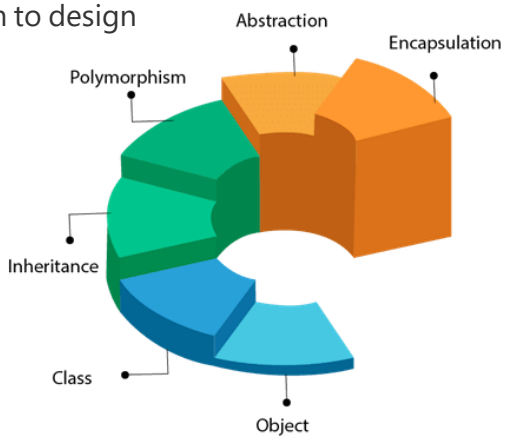
Java

Object Oriented Programming Notes

Object means a real-world entity such as a pen, chair, table, computer, watch, etc.

Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies software development and maintenance by providing some concepts:

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



Apart from these concepts, there are some other terms which are used in Object-Oriented design:

- Coupling
- Cohesion
- Association
- Aggregation
- Composition

Class

Collection of objects is called class. It is a logical entity.
A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.

Object

Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors – wagging the tail, barking, eating. An object is an instance of a class.
Example:

```
class Student {
   String name;
   int age;

   public void getInfo() {
       System.out.println("The name of this Student is " + this.name);
       System.out.println("The age of this Student is " + this.age);
   }
}

public class OOPS {
   public static void main(String args[]) {
       Student s1 = new Student();
       s1.name = "Aman";
       s1.age = 24;
       s1.getInfo();

        Student s2 = new Student();
```

Code:

```
class Student {
   String name;
   int age;

   public void displayInfo(String name) {
       System.out.println(name);
   }

   public void displayInfo(int age) {
       System.out.println(age);
   }

   public void displayInfo(String name, int age) {
       System.out.println(name);
       System.out.println(age);
   }
}
```

Runtime Polymorphism: Runtime polymorphism is also known as **dynamic polymorphism**. **Function Overriding** is an example of **runtime** polymorphism.

Function Overriding means when the child class contains the method which is already present in the parent class. Hence, **the child class overrides the method of the parent class**. In case of function overriding, parent and child classes both contain the same function with a different definition. The call to the function is determined at runtime is known as runtime polymorphism.

Code:

```
class Shape {
   public void area() {
       System.out.println("Displays Area of Shape");
   }
}
class Triangle extends Shape {
   public void area(int h, int b
       System.out.println((1/2)*b*h);
   }
}
class Circle extends Shape {
   public void area(int r) {
       System.out.println((3.14)*r*r);
   }
}
```


Abstraction

Abstraction is a process of hiding the implementation details from the user, only the functionality will be provided to the user.
In other words, the user will have the information on what the object does instead of how it does it.

In Java, abstraction is achieved using Abstract classes and interfaces.

Abstraction is achieved in 2 ways:

- Abstract class
- Interfaces (Pure Abstraction)

Abstract Class

- An abstract class must be declared with an abstract keyword.
- It can have abstract and non-abstract methods.
- It cannot be instantiated.
- It can have constructors and static methods also.
- It can have final methods which will force the subclass not to change the body of the method.

Example

```
class  Demo
{
static                //static block
{
System.out.println("Static block");
}
public static void main(String args[])  //static method
{
System.out.println("Static method");
}
}
```

```
Static block
Static method
```

We observe that JVM first executes the static block, if it is present in the program. After that it searches for the main() method. If the main() method is not found, it gives error.

Example

A program that does not have the main() method gives an error at run time.

```
class DemoStaticBlock
{
Static                          //static block
{
System.out.println("Static block");
}
}
```


Output:

```
Error: Main method not found in the class Demo, please define the main
method as:
public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application
```

So the main() method should always be written as:

```
public static void main(String args[])
```

We can interchange public and static and write it as follows:

```
static public void main(String args[])
```

We can also use the different name for the String type array and write it as:

```
static public void main(String[] x)
```

Different ways of writing main() method are:

```
static public void main(String []x)
static public void main(String...args)
```

String...args: It allows the method to accept zero or multiple arguments. There should be exactly three dots between String and array; otherwise, it gives an error.