Lecture 3: OOP principles, Object and object reference, Constructor, Types of Constructors

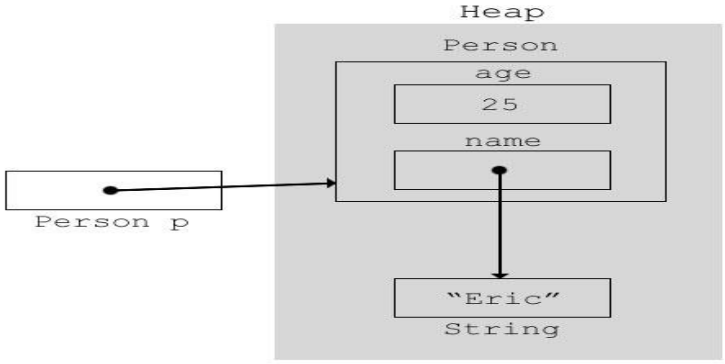
OOP principles:
Encapsulation: The wrapping up of data and function that operates on that data into a single unit is known as encapsulation.
E.g. Car brake, pedals, Mobile phone etc.

Abstraction: It is a way of hiding implementation details.

Inheritance: It is a process by which one object acquires the properties another object. With the use of inheritance the information is made manageable in a hierarchical manner.
All drivers rely on inheritance to drive different types (subclasses) of vehicles.

Polymorphism: It is a features that allows one interface to be used for a general class of actions i.e. one interface, multiple methods. So it is possible to generate an interface for a group of related activities.

Differentiate between Object and object reference:



Constructor:
A constructor initializes the internal state an object immediately upon creation.
It has the same name as the class in which it resides and is syntactically similar to a method i.e. why it is called a special method.
Every class we create has a constructor.
Once defined, the constructor is automatically called immediately after the object is created, before the new operator completes.
Constructors look a little strange because they have no return type, not even void. This is because the implicit return type of a class' constructor is the class type itself.
It is the constructor's job to initialize the internal state of an object so that the code creating an instance will have a fully initialized, usable object immediately.
E.g.

```
class MyClass{
        //This is the constructor
        MyClass(){ }
```

}

Types of Constructors
There are three types of constructors: Default, Parameter less and Parameterized.

1. Default
If we do not implement any constructor in our class, Java compiler inserts a default constructor into our code on our behalf. This constructor is known as default constructor. We should not find it in our source code (the java file) as it would be inserted into the code during compilation and exists in .class file.

2. Parameter less
Constructor with no parameter is known as parameter less constructor. The signature is same as default constructor, however body can have any code unlike default constructor where the body of the constructor is empty.
Although we may see some people claim that that default and parameter less constructor is same but in fact they are not, even if we write public MyClass() { } in our class MyClass it cannot be called default constructor since we have written the code of it.
E.g.

```
class Box {
    double width, height,depth;
    // This is the constructor for Box.
    Box(){width=1; height=2; depth=3;}
    public static void main(String args[]) {
            Box mybox1 = new Box();
            Box mybox2 = new Box();
    }
}
```

3. Parameterized
Constructor with parameters is known as Parameterized constructor. Box( ) constructor in the preceding example does initialize a Box object, it is not very useful—all boxes have the same dimensions. What is needed is a way to construct Box objects of various dimensions. The easy solution is to add parameters to the constructor.

E.g.

```
class Box {
    double width, height,depth;
    // This is the constructor for Box.
    Box(double w,double h,double d){width=w; height=h; depth=d;}
    public static void main(String args[]) {
            Box mybox1 = new Box(1,2,3);
            Box mybox2 = new Box(2,3,4);
    }
}
```