MAJOR OOP CONCEPTS

December 21, 2015

.

- Inheritance - Abstraction - IS-A / HAS-A
- PolyMorphism - Dynamic Binding
- Overriding
- Overloading
- Abstract Classes
- Interfaces

1 Inheritance

$\rightarrow$ *The concept of Inheritance is fundamental to Object Orientation.*

$\rightarrow$ *The concept is based on the fact that, we will abstract out all the features of subclasses and place them inside the superclass.*

eg 1: A car has the following properties. .

- mileage
- engine - capacity
- occupancy
- purpose

So, a car can be modelled in java like

```
Class Car
 {
    int occupancy;
    int engine-capacity;
    float mileage;
    void purpose()
       {
        System.Out.Println (”Family trip or Cab Service”);
      }
    void HatchbackOrSedan ()
      {
        System.Out.Println (”HatchbackOrSedan ”);
      }
  }
```

Above program follows exclusive to Cars

Similarly a bike can be modelled as follows

```
Class Bike
  {
    int occupancy;
    int engine-capacity;
    float mileage;
    void purpose()
      {
        System.Out.Println (”Go out with friends or GF”);
      }
    void GearOrWithoutGear ()
      {
        System.Out.Println (” Geared / Without Geared”);
      }
  }
```

Above program follows exclusive to bike

$\rightarrow$ *So in the above illustration, we can identify that..*

- mileage
- engine - capacity
- occupancy
- purpose()

Above properties common for both CAR and BIKE

HatchbackOrSedan () $\rightarrow EXCLUSIVE\ TO\ CAR$
GearOrWithoutGear () $\rightarrow EXCLUSIVE\ TO\ BIKE$

$\rightarrow$ *When you find such scanacious, we will abstract out all the common features and place them inside a superclass.*

$\rightarrow$ *This process is called "abstraction".So the above illustration can be modelled as follows.*

```
    Class Vehicle
     {
        int occupancy;
        int engine-capacity;
        float mileage;
Above lines are for Abstraction
        void purpose()
          {
             System.Out.Println ("Some purpose");
          }
     }
  Class Car extends Vehicle
          {
        void purpose()
          {
             System.Out.Println ("Family trip or Cab Service");

             }
        void HatchbackOrSedan ()
          {
             System.Out.Println ("HatchbackOrSedan ");
          }
     }
  Class Bike extends Vehicle
          {
        void purpose()
          {
             System.Out.Println ("Go out with friends or GF");
        }
      void GearOrWithoutGear ()
        {
          System.Out.Println (" Geared / Without Geared");
        }
    }
```

2 Observations

→ *Variables and methods which are common to all subclasses are taken out and placed in the superclass,"vehicle".*

→ *Method purpose() is performing different things in Car and Bike subclasses.So you abstracted out and kept in the superclass.You gave some "dummy" body to it as following :*

```
void purpose()
{
  System.Out.Println ("Some purpose");
}
```

Above program is dummy body given in superclass vehicle

→ *You later overrided in subclasses.*
→ *The class specific properties such as*

HatchbackOrSedan () → *EXCLUSIVE TO CAR*

GearOrWithoutGear () → *EXCLUSIVE TO BIKE*

ARE NOT PLACED IN THE SUPERCLASS.

→ *Now we will extend it to class Animal and lets see,how inheritance works for our abstraction.*

Now let us observe the Dog,Cat,Horse and abstract the features to a animal.

```
Class Dog
 {
    int weight;
    int colour;
    int age;
    void talk()
```

```
        {
            System.Out.Println ("bow");
        }
    void eat()
        {
            System.Out.Println ("bones");
        }
    void provide Security()
        {
            System.Out.Println ("Security to houses");
        }
}
```

Above program is exclusive to Dog Class

```
Class Cat
{
    int weight;
    int colour;
    int age;
    void talk()
        {
            System.Out.Println ("Meow");
        }
    void eat()
        {
            System.Out.Println ("drinking Milk");
        }
    void StealMilk()
        {
            System.Out.Println ("Steal the Milk");
        }
}
```

Above program is exclusive to Cat Class

```
Class Horse
 {
    int weight;
    int colour;
    int age;
    void talk()
       {
          System.Out.Println (”i”);
       }
    void eat()
       {
          System.Out.Println (”Eating Grass”);
       }
    void transport()
       {
          System.Out.Println (”Pulling Carts”);
       }
 }
```

→ *Let us abstract the common features out and keep it inside Animal.*

```
Class Horse
 {
    int weight;
    int colour;
    int age;
    void eat()
       {
          System.Out.Println (”Eat Something”);
       }
    void talk()
       {
          System.Out.Println (”Talking”);
       }
 }
```

All these things are common to Dog,Cat,Horse.So abstracted them to Animal

```
  Class Cat extends Animal
   {
      void talk()
         {
            System.Out.Println (”Meow”);
         }
      void eat()
         {
            System.Out.Println (”drinking Milk”);
         }
      void StealMilk()
         {
            System.Out.Println (”Steal the Milk”);
         }
   }
Class Dog extends Animal
   {
      void talk()
         {
            System.Out.Println (”bow”);
         }
      void eat()
         {
            System.Out.Println (”bones”);
         }
      void provide Security()
         {
            System.Out.Println (”Security to houses”);
         }
   }
Class Horse extends Animal
   {
        void talk()
          {
            System.Out.Println ("i");
          }
        void eat()
          {
            System.Out.Println ("Eating Grass");
          }
        void transport()
          {
            System.Out.Println ("Pulling Carts");
          }
      }
```

→ *After understanding what is inheritance,let us see some examples how to use them.*