IT-2204b

1) Interface Vs Abstruction

⇒ Interface:

Interface is like a contract which must be fullfilled entirely. We can use interface some strict Rules must be followed.

Now consider a scenario for example a Interface called Tea_maker has 5 abstract method like

```
bringwater();
boilwater();
:
:
```

Now a class SelfMan Must implement all the method of TeaMaker interface To Make Serve a cup of Tea. Missing one or two method then It is not possible to make a cup of Tea.

IT-22046

Code:

```
interface TeaMaken {
    void bringWater();
    void boilwater();
    void add_TeaBag();
    void addSugar();
    void serveTea();
}

class SellsMan implements TeaMaken {
    int sugarAmount;
    public void set_sugar amount (int SugarAm) {
        this.sugarAmount = SugarAm;
    }
    public void bringWater() {
        System.out.println("BringingWater");
    }
    public void boilwater() {
        System.out.println("BoilingWater");
    }
```

IT-22046

```
public void addTeaBag() {
    System.out.println("Adding Tea Bag");
}
public void addSugar() {
    System.out.println("Added sugar" +
        this.sugarAmount + "tea spoo");
}
public void serveTea() {
    System.out.println("Take your Tea...");
}
}

Public class TestInterface {
    public void static void main(String[] args) {
        SellsMan tea = New SellsMan();
        tea.bringWater();
        tea.boilWater();
        tea.addTeaBag();
        tea.setSugarAmount(2);
        tea.addSugar();
        tea.serve_Tea();
    }
}
```

IT-22046

⇒ Abstraction:

abstract classes one the ~~and~~ Class which can contain both abstract an concrete method.

Now a consider a scenario that a painter uses colour to paint. He use Red, blue etc. colour ~~##~~ in common but some times he needs to ~~##~~ make other colours as per demand.

So let's create a abstract class 'Colours' which will contain void red(); void blue(); as concrete class, as they are being used often.

IT-22046

Also make a abstract method called 'makeYourOwnColour()' to make the custom colour.

code:

```
abstract class colours {
    int red, green, blue;
    abstract void makeYourOwn(int red,
                              int green, int blue);
    void red() {
        System.out.println("Red painted");
    }
    void blue() {
        System.out.println("Blue painted");
    }
}
```

IT-22046

```
class Painter extends Colours {

    void makeYourOwn(int red, int green
                     , int blue) {
        this.red = red;
        this.green = green;
        this.blue = blue;
        System.out.println("New colour
        painted with red:"+red+", green+...)
    }
}

public class Test_Abstruction {
    public static void main(String[]args){
        Painter p= new Painter();
        P.red();
        P.blue();
        P.makeYourOwnColour(100,150,200);
    }
}
```