```java
package Questions;

final class NewClass{
    int a;
    int b;
}
public class FinalClass extends NewClass{
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}

```

```java
package Questions;

class FinalShape{
    public final void print(){
        System.out.println("This is a final shape");
    }
}
public class FinalMethods extends FinalShape{
    public void print(){
        System.out.println("This is inherited shape");
    }

    public static void main(String[] args) {
        FinalMethods fm = new FinalMethods();
        fm.print();
    }
}

```

```java
package Questions;

abstract class AbstractShape{ // Private member variable private
     String color;
    // Constructor
    public AbstractShape (String color) {
        this.color = color;
    }
    public String toString() {
        return "Shape of color=\"" + color + "\"";
    }
    // All Shape subclasses must implement a method called getArea()
    abstract public double getArea();
}
public class AbstractClass {
    public static void main(String[] args) {
            AbstractShape s1 = new Rectangle("red", 4, 5);
            System.out.println(s1);
            System.out.println("Area is " + s1.getArea());
            AbstractShape s2 = new Triangle("blue", 4, 5);
            System.out.println(s2);
            System.out.println("Area is " + s2.getArea());
// Cannot create instance of an abstract class
            AbstractShape s3 = new AbstractShape("green");// Compilation Error!!
    }
}

```

```java
package Questions;

class Shape {
    private String color;
    public Shape (String color) {
        this.color = color;
    }
    public String toString() {
        return "Shape of color=\"" + color + "\""; }
    public double getArea() {
        System.err.println("Shape unknown! Cannot compute area!"); return 0; // Need a return to compile the program
    } }

class Rectangle extends Shape { // Private member variables
    private int length;
    private int width;
    public Rectangle(String color, int length, int width) {
        super(color);
        this.length = length;
        this.width = width;
    }
    public String toString() {
        return "Rectangle of length=" + length + " and width=" + width + ", subclass of " + super.toString(); }
        public double getArea() { return length*width; }
}

class Triangle extends Shape { // Private member variables private int base;
    private int height;
    private int base;
    // Constructor
    public Triangle(String color, int base, int height) {
        super(color);
        this.base = base;
        this.height = height;
    }
    public String toString() {
        return "Triangle of base=" + base + " and height=" + height + ", subclass of " + super.toString();
    }
    public double getArea() { return 0.5*base*height; }
}


public class MethodOverriding {
    public static void main(String[] args) {
        Shape s1 = new Rectangle("red", 4, 5);
        System.out.println(s1);
        System.out.println("Area is " + s1.getArea());
        Shape s2 = new Triangle("blue", 4, 5);
        System.out.println(s2);
        System.out.println("Area is " + s2.getArea());
    }
}

```

```java
package Questions;
//exercise 1.2
public class SingleInheritanceMain {
    public static void main(String[] args) {
        E e = new E();
        e.show();
    }
}

class A{
    public int a =100;
}

class B extends A{
    public int a = 80;
}
class C extends B{
    public int a = 60;
}

class D extends C{
    public int a = 40;
}

class E extends D{
    public int a = 10;
    public void show(){
        System.out.println("E: "+this.a);
        System.out.println("D: "+((D) this).a);
        System.out.println("C: "+((C) this).a);
        System.out.println("B: "+((B) this).a);
        System.out.println("A: "+((A) this).a);
    }
}

```

```java
package Questions;


class Room {
    double length,breadth,height;
    public Room(){
        length=-1;
        breadth=-1;
        height=-1;
    }

     public Room(double l,double b,double h) {
        length=l;
        breadth=b;
        height=h;
    }
    Room(double len) { // Single parametrised constructor
         length=breadth=height=len;
    }

    double volume() {
        return length*breadth*height;
    }
}
public class ConstructorOverloading {
    public static void main(String args[]) {
        Room a = new Room(20, 30, 40);
        Room b = new Room();
        Room c = new Room(10);
        double vol;
        vol = a.volume();
        System.out.println("Volume of room a is " + vol);
        vol = b.volume();
        System.out.println("Volume of room b is " + vol);
        vol = c.volume();
        System.out.println("Volume of room c is " + vol);
    }
}

```