Ejercicios

1. Cuales son las ventajas de usar herencia?
2. Cual es la diferencia entre this y super?
3. A que se refiere la sobrecarga de métodos? Escriba un ejemplo.
4. A que se refiere la sobre-escritura de métodos? Escriba un ejemplo.

5. Que imprime el siguiente código?

```
A a1= new A();
A a2= new B();
A a3= new B();
System.out.println(a1.bar());
System.out.println(a2.bar());
System.out.println(b.bar());

class A{
      public int bar(){ return this.getValue();}
      public int getValue(){return 2;}
}
class B{
      public int getValue(){return 4;}
      public int bar(){return super.bar()*2;}
}
```

6. Que imprime el siguiente código?

```
ColoredPoint point1 = new ColoredPoint(1, 2, Color.GRAY);
Point point2 = new ColoredPoint(2,3,Color.GREEN);
System.out.println(point1.toString());
System.out.println(point2.toString());

class Point{
      int x,y;
      Point(int x, int y){
            this.x=x;
            this.y=y;
      }
      String toString(){
            return "Point("+ this.x + ","+ this.y+")";
      }
}
class ColoredPoint extends Point{
      private Color color;
      ColoredPoint(int x, int y, Color aColor){
            super(x,y);
            this.color=color;
      }
      String toString(){
            return "Colored"+ super.toString();
      }
}
```

7. Considere las siguientes clases que modelan 2 tipos de cuentas bancarias.

- **TransactionAccount:** que a fin de mes descuenta 0.25 dolares por cada transacción realizada durante el mes.
- **FeeAccount:** que a fin de mes descuenta un monto fijo de 5 dolares, sin importar el numero de transacciones.

```
public class TransactionAccount{
   protected double balance;
   protected int transactions;
   public TransactionAccount(double balance) {
      this.balance = balance;
      transactions = 0;
   }
   public void withdraw(double amt) {
      balance = balance - amt;
      transactions++;
   }
   public void deposit(double amt) {
      balance = balance + amt;
      transactions++;
   }
   public double getBalance() { return balance; }
   public void endMonth() {
      endMonthCharge();
      transactions = 0;
   }
   protected void endMonthCharge(){
         balance = balance - transactions * 0.25;
   }
}
public class FeedAccount{
   protected double balance;
   protected int transactions;
   public FeedAccount(double balance) {
      this.balance = balance;
      transactions = 0;
   }
   public void withdraw(double amt) {
      balance = balance - amt;
      transactions++;
   }
   public void deposit(double amt) {
      balance = balance + amt;
      transactions++;
   }
   public double getBalance() { return balance; }
   public void endMonth() {
      endMonthCharge();
      transactions = 0;
   }
   protected void endMonthCharge(){
         balance = balance - 5.0;

   }
}
```

Que opina del código anterior? Proponga una mejor solución que permita modelar e implementar los dos tipos de cuenta de una manera mas apropiada. **Escriba código.**