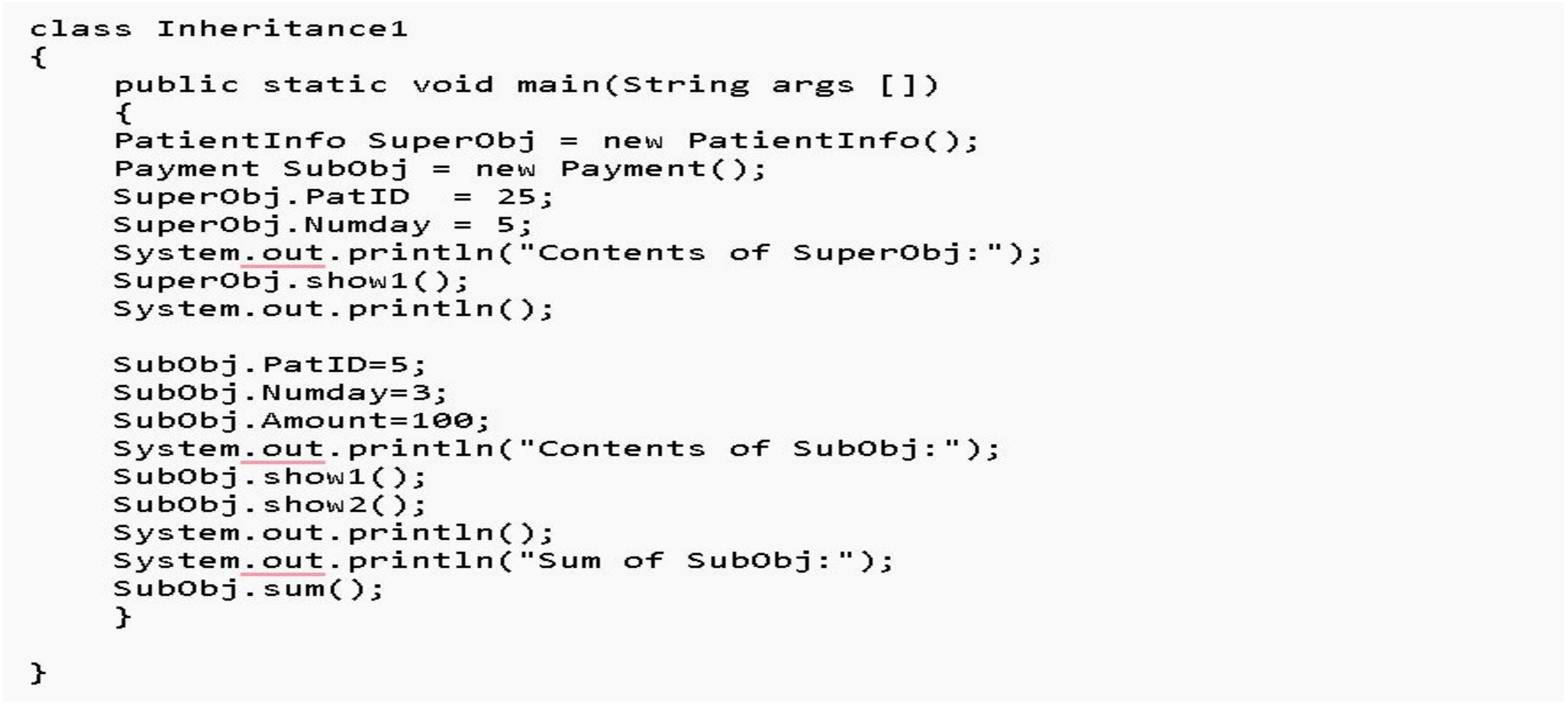
UNIT - II

INHERITANCE

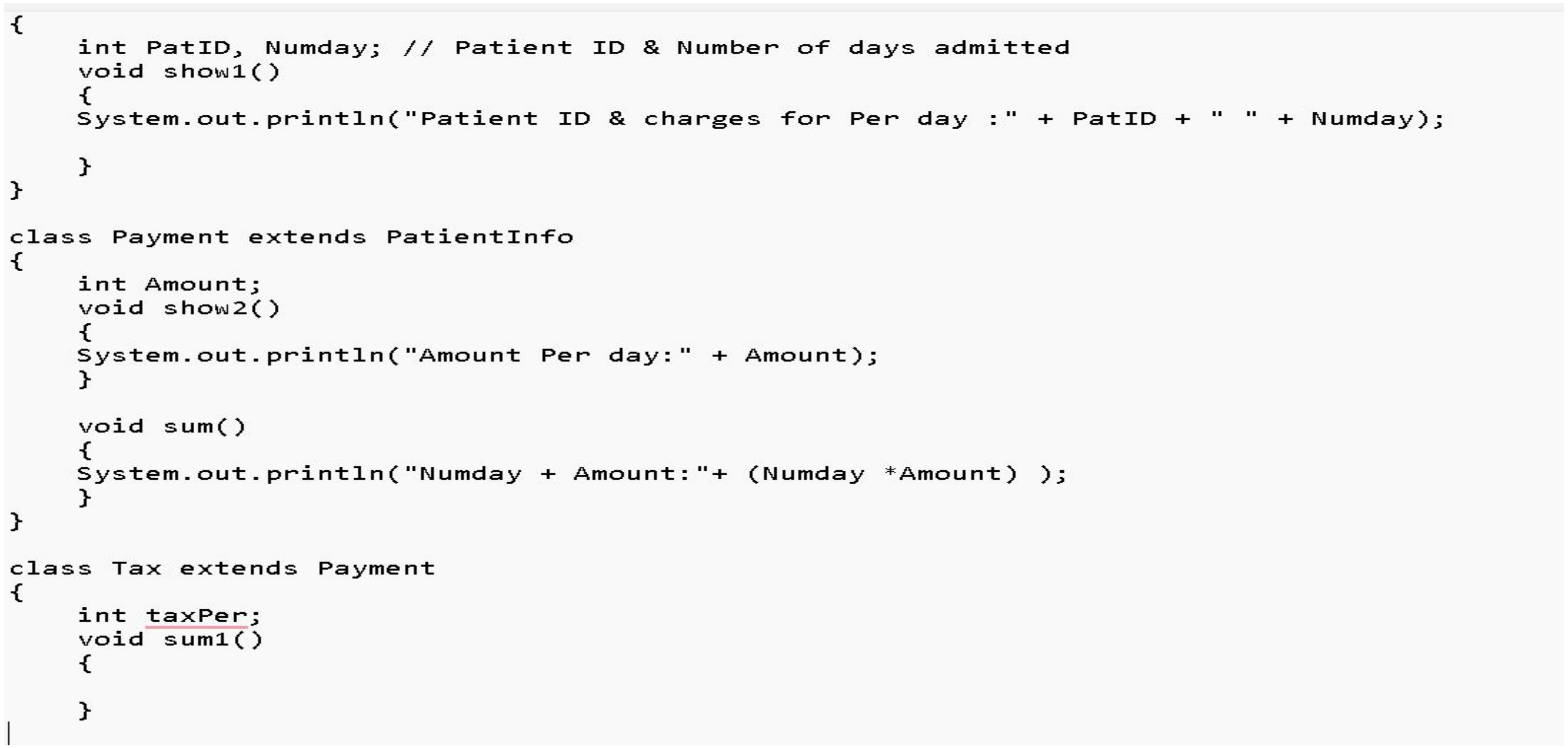
Sample Program for Inheritance Concept

```
class PatientInfo
{
    int PatID, Numday; // Patient ID & Number of days admitted
    void show1()
    {
    System.out.println("Patient ID & charges for Per day :" + PatID + " " + Numday);

    }
}

class Payment extends PatientInfo
{
    int Amount;
    void show2()
    {
    System.out.println("Amount Per day:" + Amount);
    }

    void sum()
    {
    System.out.println("Numday + Amount:"+ (Numday *Amount) );
    }
}
```



```
class Inheritance1
{
    public static void main(String args [])
    {
    PatientInfo SuperObj = new PatientInfo();
    Payment SubObj = new Payment();
    SuperObj.PatID  = 25;
    SuperObj.Numday = 5;
    System.out.println("Contents of SuperObj:");
    SuperObj.show1();
    System.out.println();

    SubObj.PatID=5;
    SubObj.Numday=3;
    SubObj.Amount=100;
    System.out.println("Contents of SubObj:");
    SubObj.show1();
    SubObj.show2();
    System.out.println();
    System.out.println("Sum of SubObj:");
    SubObj.sum();
    }

}
```

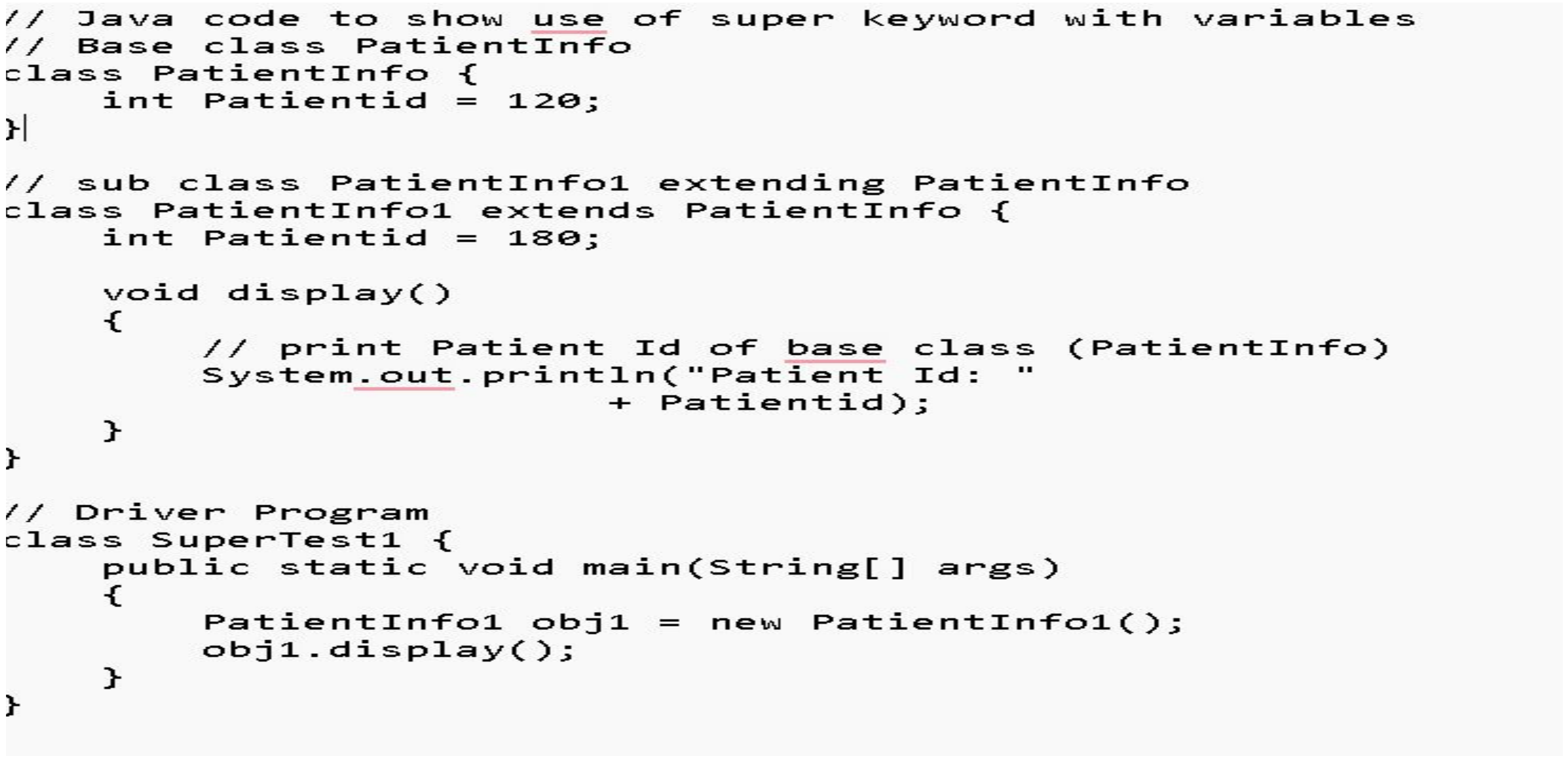
Multilevel Hierarchy



```
{
    int PatID, Numday; // Patient ID & Number of days admitted
    void show1()
    {
    System.out.println("Patient ID & charges for Per day :" + PatID + " " + Numday);

    }
}

class Payment extends PatientInfo
{
    int Amount;
    void show2()
    {
    System.out.println("Amount Per day:" + Amount);
    }

    void sum()
    {
    System.out.println("Numday + Amount:"+ (Numday *Amount) );
    }
}

class Tax extends Payment
{
    int taxPer;
    void sum1()
    {

    }
```

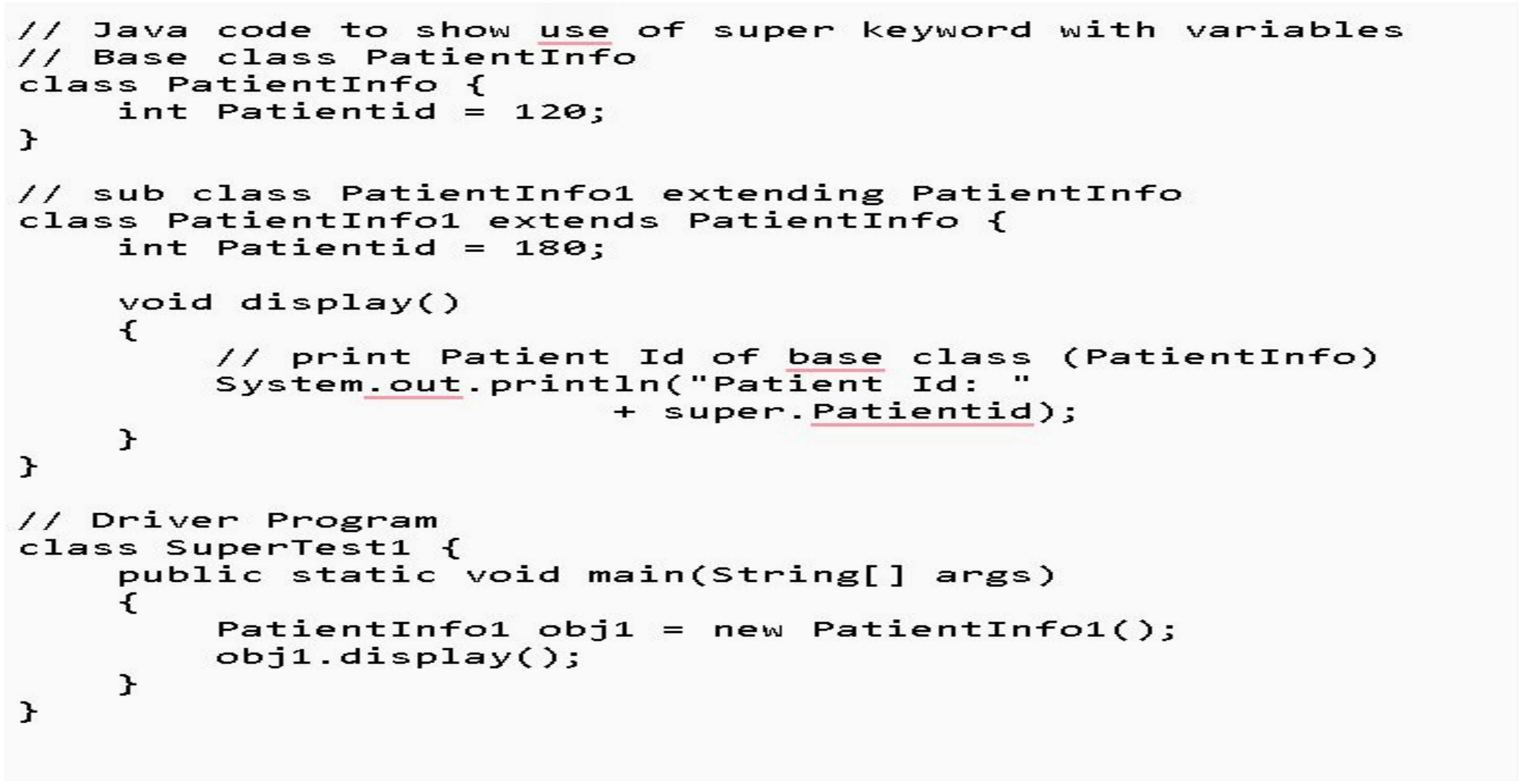
Using super

- The super keyword in Java is a reference variable which is used to refer immediate parent class object.
- Whenever we create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.
- super Keyword used in three Different types
    - super can be used to refer immediate parent class instance variable.
    - super can be used to invoke immediate parent class method.
    - super() can be used to invoke immediate parent class constructor.



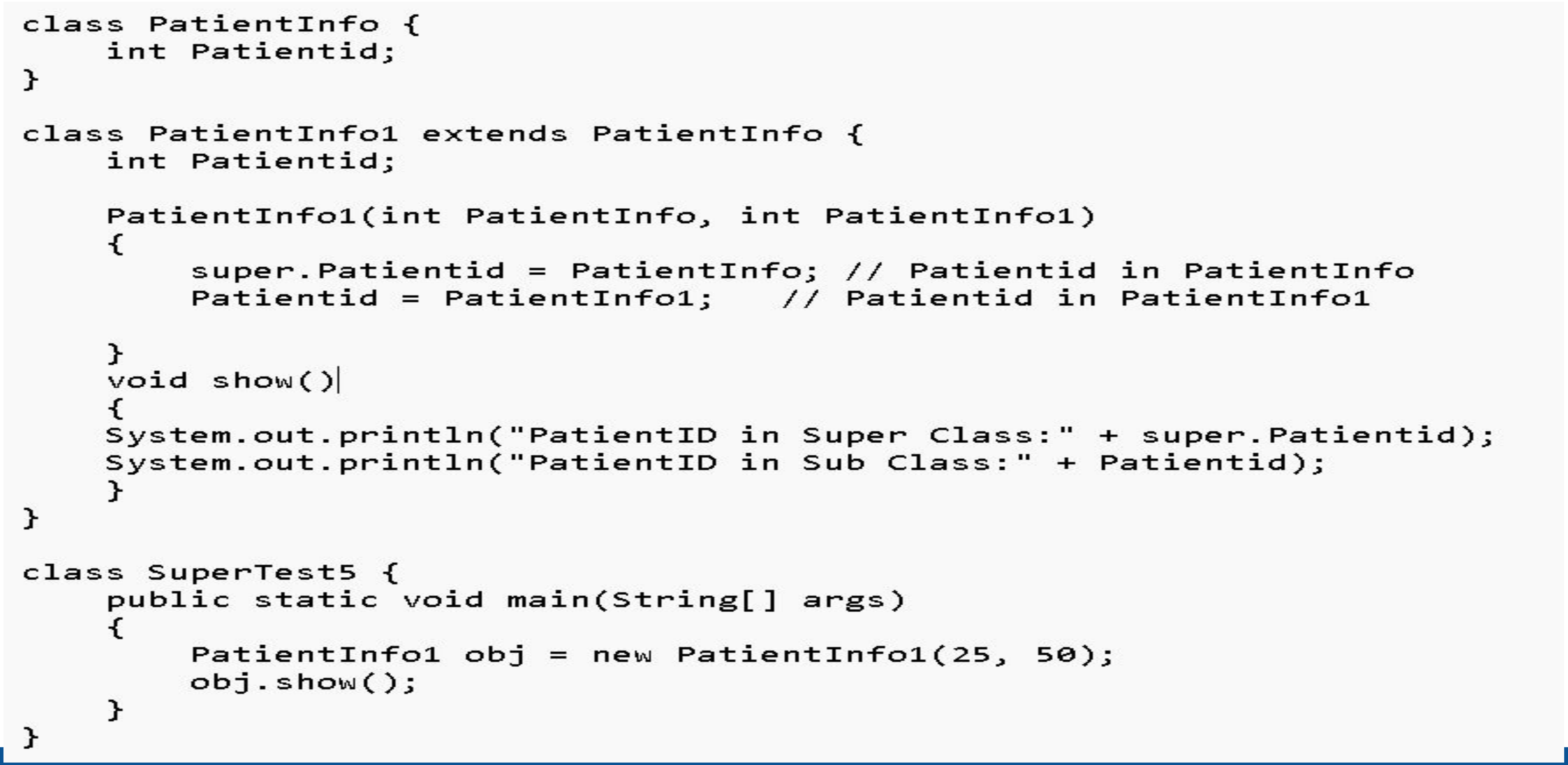
```
// Java code to show use of super keyword with variables
// Base class PatientInfo
class PatientInfo {
    int Patientid = 120;
}

// sub class PatientInfo1 extending PatientInfo
class PatientInfo1 extends PatientInfo {
    int Patientid = 180;

    void display()
    {
        // print Patient Id of base class (PatientInfo)
        System.out.println("Patient Id:  "
                        + Patientid);
    }
}

// Driver Program
class SuperTest1 {
    public static void main(String[] args)
    {
        PatientInfo1 obj1 = new PatientInfo1();
        obj1.display();
    }
}
```

Use of super with variables



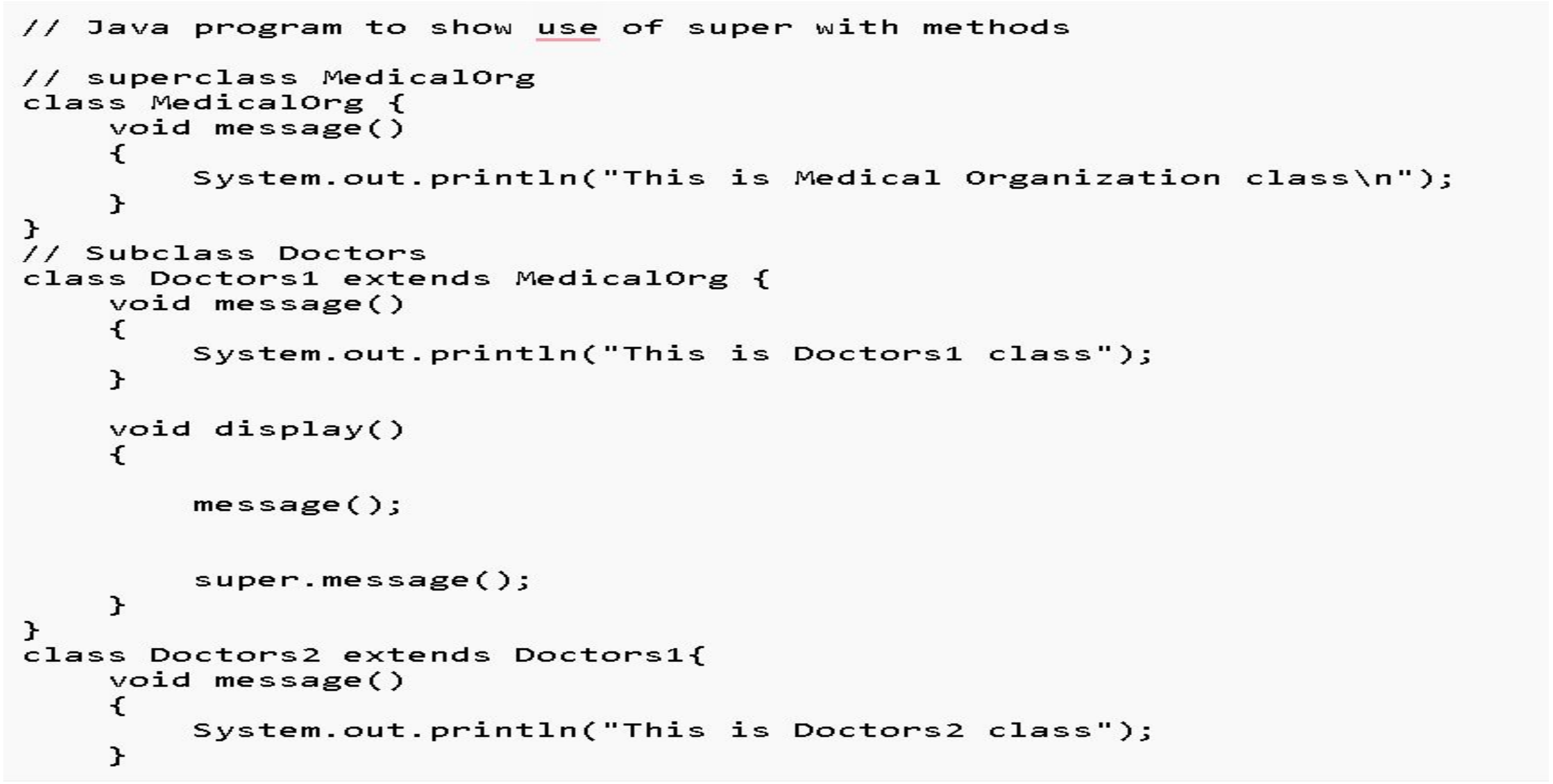
```
// Java code to show use of super keyword with variables
// Base class PatientInfo
class PatientInfo {
    int Patientid = 120;
}

// sub class PatientInfo1 extending PatientInfo
class PatientInfo1 extends PatientInfo {
    int Patientid = 180;

    void display()
    {
        // print Patient Id of base class (PatientInfo)
        System.out.println("Patient Id: "
                        + super.Patientid);
    }
}

// Driver Program
class SuperTest1 {
    public static void main(String[] args)
    {
        PatientInfo1 obj1 = new PatientInfo1();
        obj1.display();
    }
}
```

Sample Program



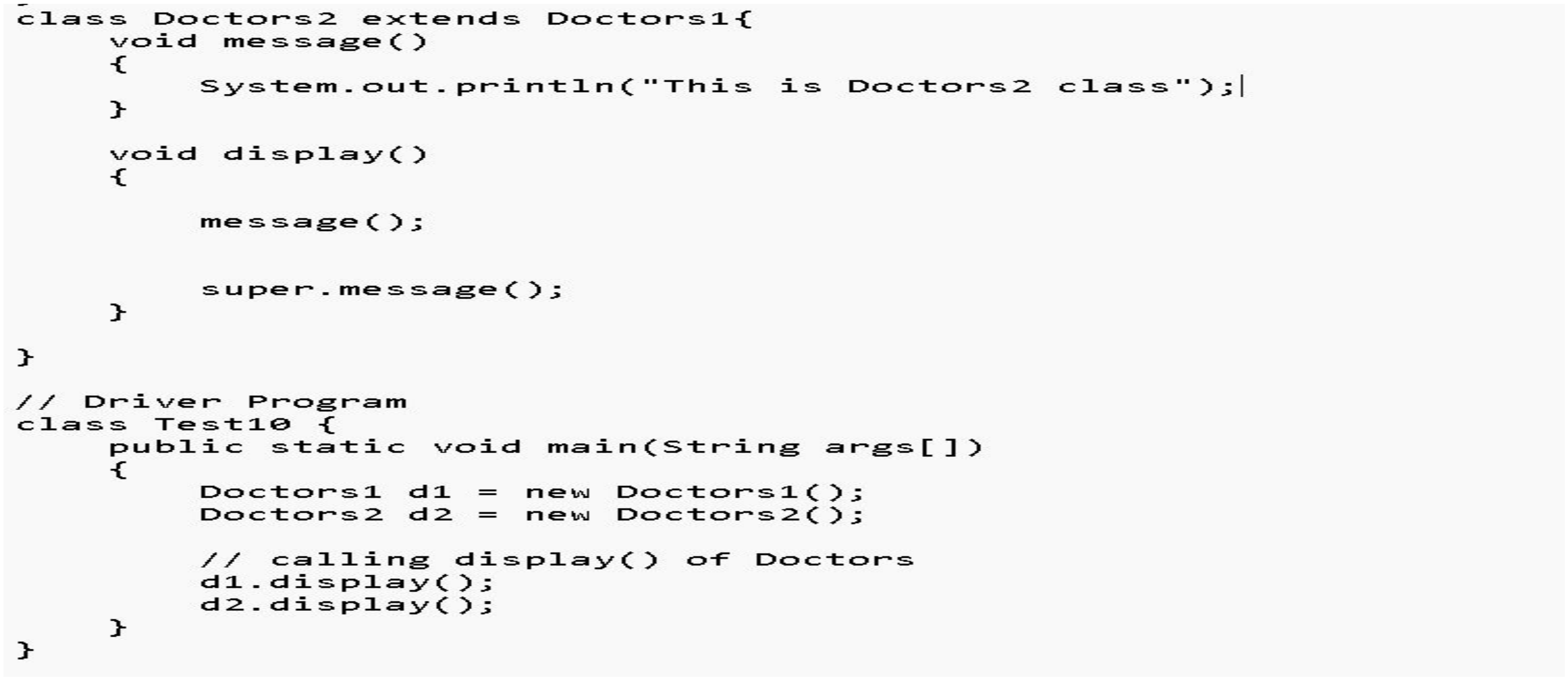
```java
class PatientInfo {
    int Patientid;
}

class PatientInfo1 extends PatientInfo {
    int Patientid;

    PatientInfo1(int PatientInfo, int PatientInfo1)
    {
        super.Patientid = PatientInfo; // Patientid in PatientInfo
        Patientid = PatientInfo1;   // Patientid in PatientInfo1

    }
    void show()
    {
    System.out.println("PatientID in Super Class:" + super.Patientid);
    System.out.println("PatientID in Sub Class:" + Patientid);
    }
}

class SuperTest5 {
    public static void main(String[] args)
    {
        PatientInfo1 obj = new PatientInfo1(25, 50);
        obj.show();
    }
}
```

Use of super with Method



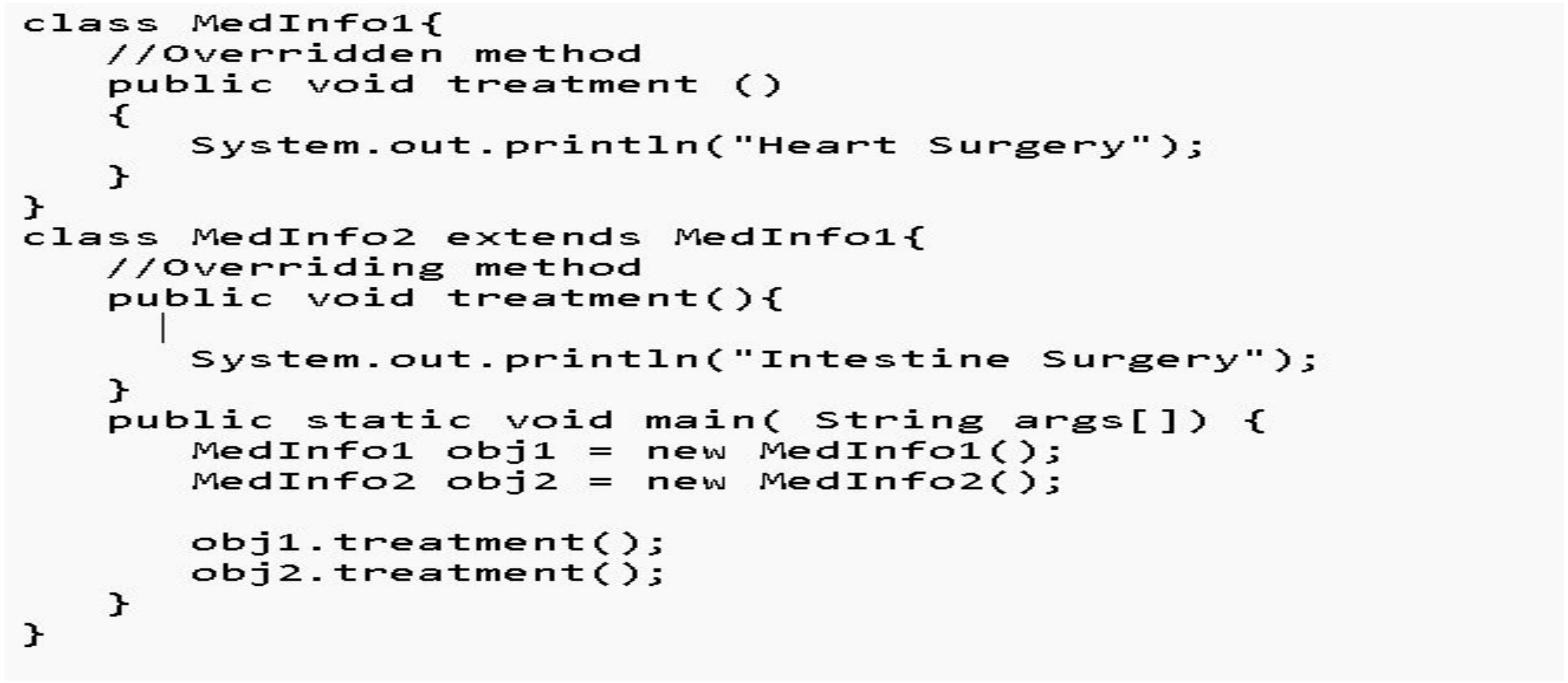
```
// Java program to show use of super with methods

// superclass MedicalOrg
class MedicalOrg {
    void message()
    {
        System.out.println("This is Medical Organization class\n");
    }
}
// Subclass Doctors
class Doctors1 extends MedicalOrg {
    void message()
    {
        System.out.println("This is Doctors1 class");
    }

    void display()
    {

        message();


        super.message();
    }
}
class Doctors2 extends Doctors1{
    void message()
    {
        System.out.println("This is Doctors2 class");
    }
```

Use of super with Method



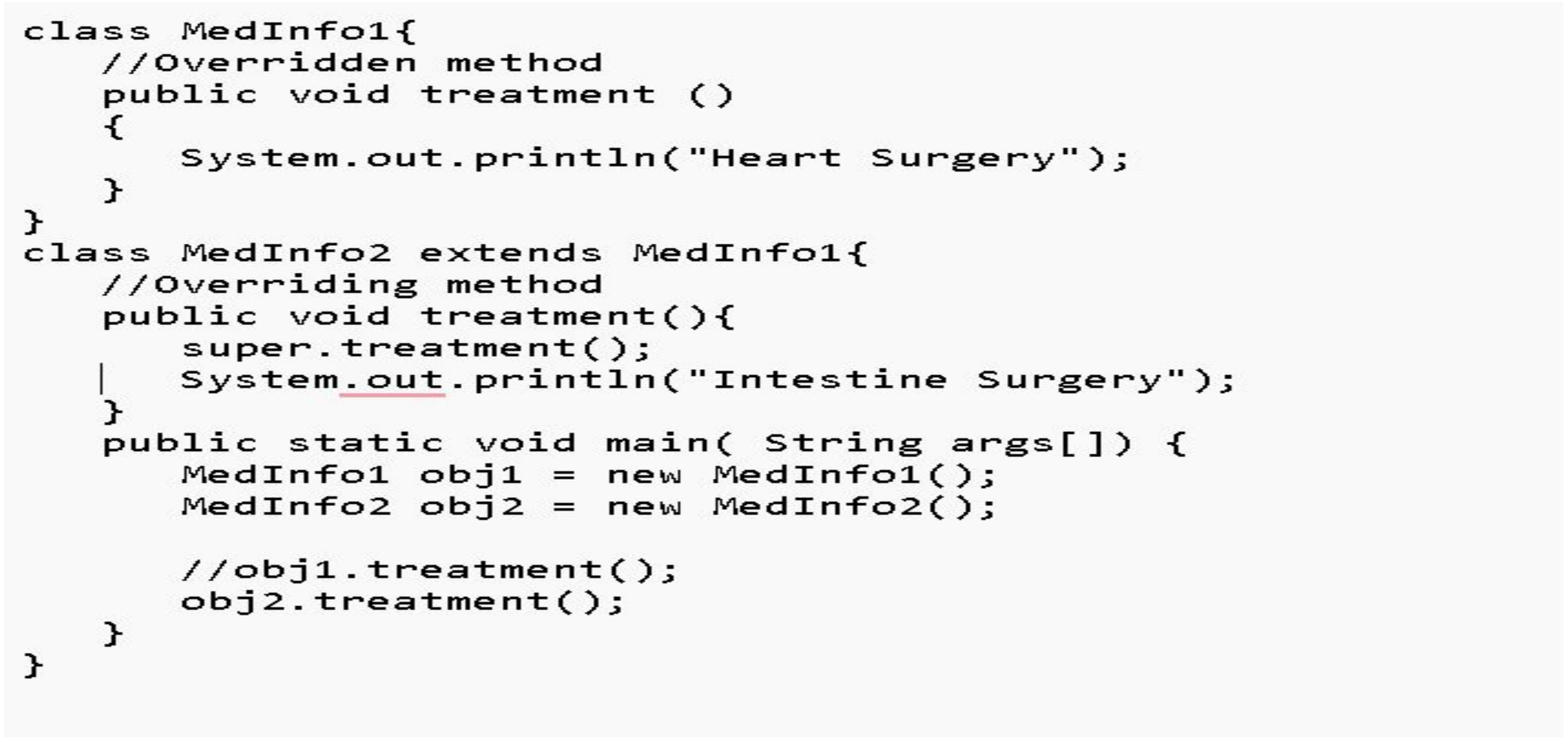
```
class Doctors2 extends Doctors1{
    void message()
    {
        System.out.println("This is Doctors2 class");
    }

    void display()
    {

        message();


        super.message();
    }

}

// Driver Program
class Test10 {
    public static void main(String args[])
    {
        Doctors1 d1 = new Doctors1();
        Doctors2 d2 = new Doctors2();

        // calling display() of Doctors
        d1.display();
        d2.display();
    }
}
```

Method overriding



```
class MedInfo1{
   //Overridden method
   public void treatment ()
   {
      System.out.println("Heart Surgery");
   }
}
class MedInfo2 extends MedInfo1{
   //Overriding method
   public void treatment(){

      System.out.println("Intestine Surgery");
   }
   public static void main( String args[]) {
      MedInfo1 obj1 = new MedInfo1();
      MedInfo2 obj2 = new MedInfo2();

      obj1.treatment();
      obj2.treatment();
   }
}
```

Method overriding using super keyword



```
class MedInfo1{
    //Overridden method
    public void treatment ()
    {
        System.out.println("Heart Surgery");
    }
}
class MedInfo2 extends MedInfo1{
    //Overriding method
    public void treatment(){
        super.treatment();
        System.out.println("Intestine Surgery");
    }
    public static void main( String args[]) {
        MedInfo1 obj1 = new MedInfo1();
        MedInfo2 obj2 = new MedInfo2();

        //obj1.treatment();
        obj2.treatment();
    }
}
```

Difference between Overloading & Overriding

Overloading	Overriding
Implements "compile time polymorphism"	Implements "runtime polymorphism"
The method call is determined at compile time	The method call is determined at runtime based on the object type
Occurs between the methods in the same class	Occurs between superclass and subclass

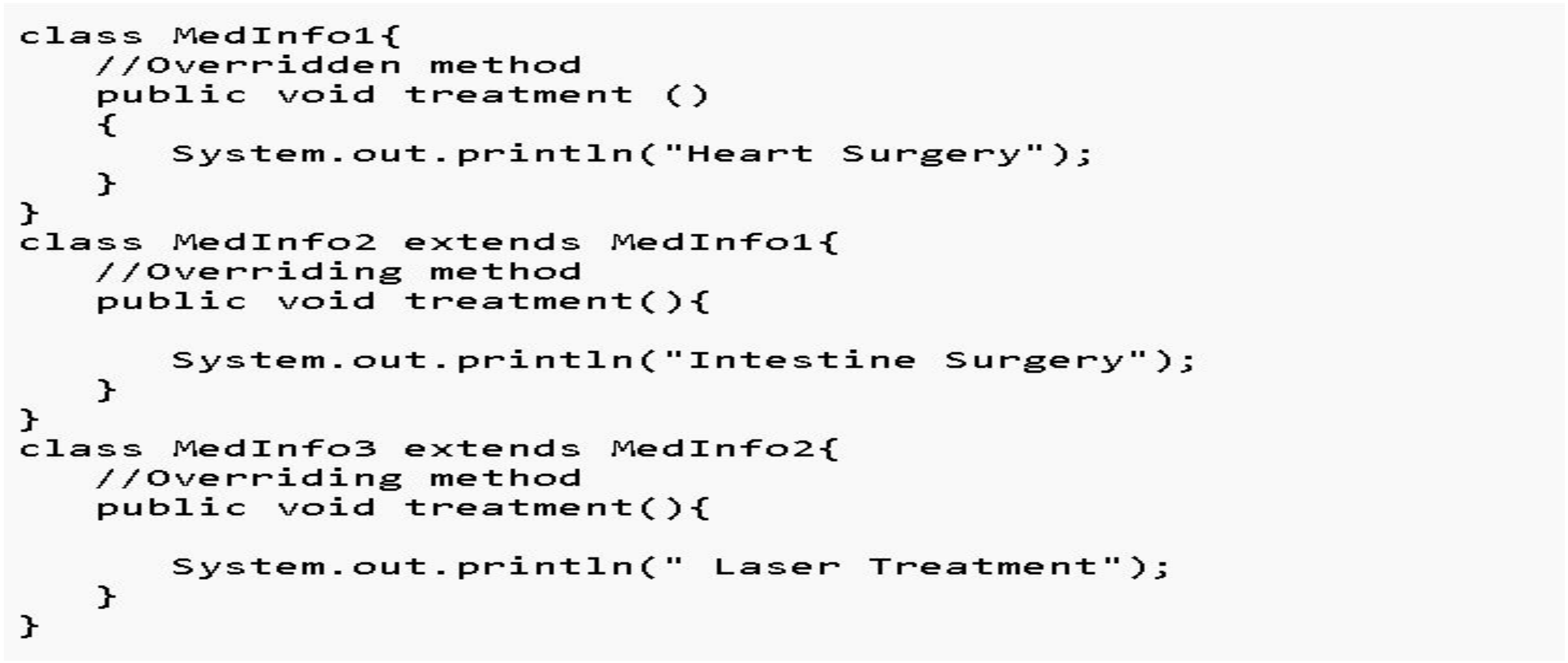
Dynamic Method Dispatch

- Method overriding forms the basis for one of Java’s most powerful **concepts: dynamic method dispatch.**

- Dynamic method dispatch is the mechanism by which a call to an overridden **method is resolved at run time,** rather than compile time.

- Dynamic method dispatch is important because this is **how Java implements run-time polymorphism.**

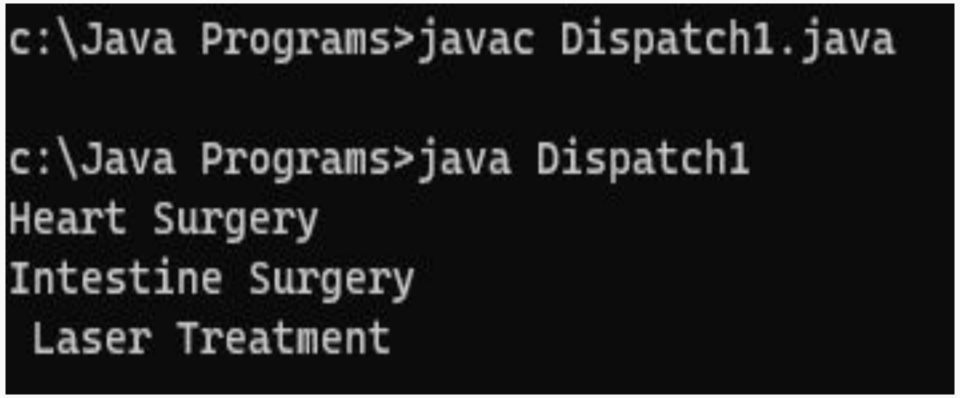
Dispatch Program



```
class MedInfo1{
    //Overridden method
    public void treatment ()
    {
        System.out.println("Heart Surgery");
    }
}
class MedInfo2 extends MedInfo1{
    //Overriding method
    public void treatment(){

        System.out.println("Intestine Surgery");
    }
}
class MedInfo3 extends MedInfo2{
    //Overriding method
    public void treatment(){

        System.out.println(" Laser Treatment");
    }
}
```

Dispatch Program

```
class Dispatch1 {
   public static void main( String args[]) {
      MedInfo1 obj1 = new MedInfo1();
      MedInfo2 obj2 = new MedInfo2();
      MedInfo3 obj3 = new MedInfo3();

    MedInfo1 Dispatch; // Obtain a reference of type MedInfo1

    Dispatch = obj1; // Dispatch referes to an MedInfo1 object
    Dispatch.treatment(); // calls MedInfo1 of treatment

    Dispatch = obj2;
    Dispatch.treatment();

    Dispatch = obj3;
    Dispatch.treatment();

   }
}
```



```
c:\Java Programs>javac Dispatch1.java

c:\Java Programs>java Dispatch1
Heart Surgery
Intestine Surgery
 Laser Treatment
```

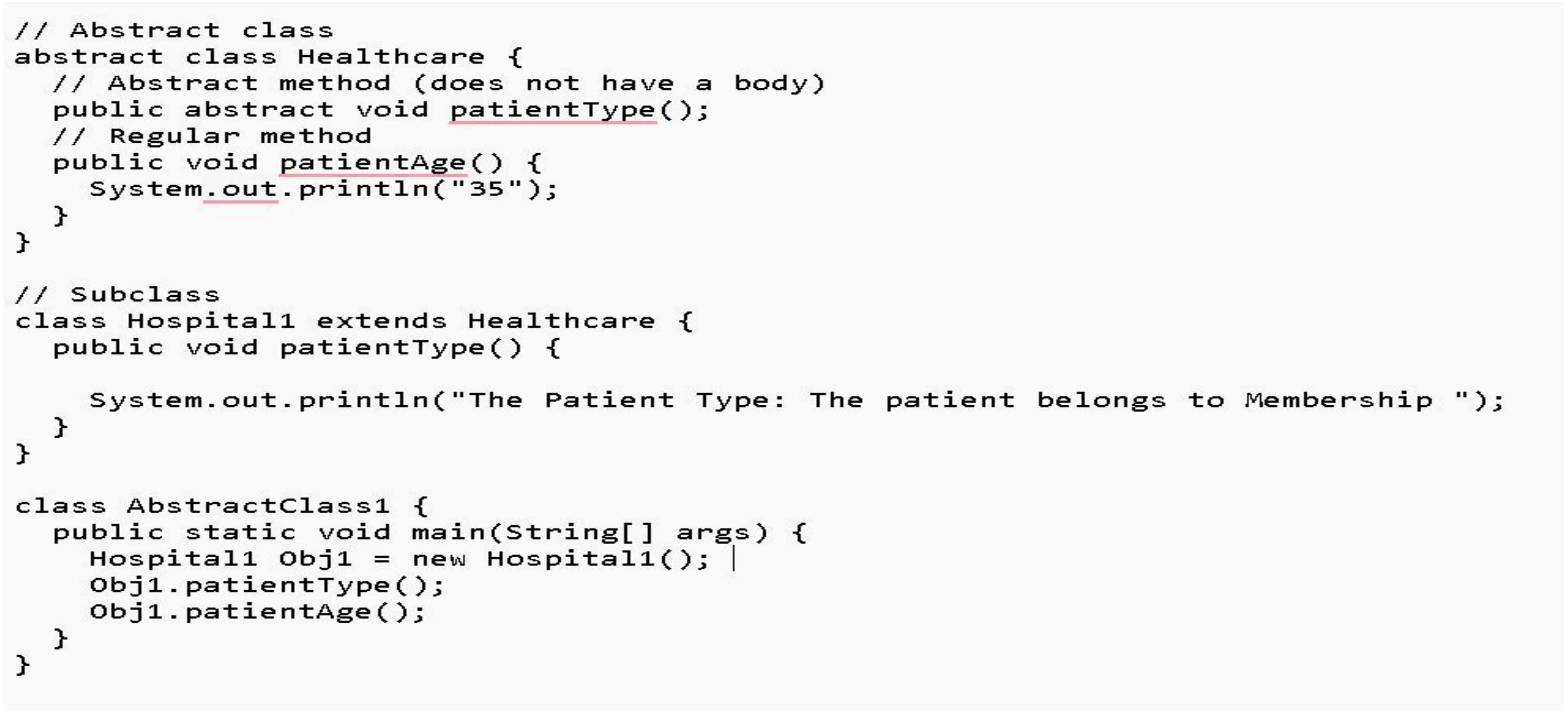
Abstract keyword

- An abstract class in Java is one that is declared with the abstract keyword.
- It may have both abstract and non-abstract methods(methods with bodies).
- An abstract is a java modifier applicable for classes and methods in java but not for Variables.

Points to Remember

- An abstract class must be declared **with an abstract keyword.**
- It can have abstract and non-abstract methods.
- It cannot be instantiated.
- It can have constructors and static methods also.
- It can have final methods which will force the subclass not to change the body of the method.

Sample Program



```
// Abstract class
abstract class Healthcare {
  // Abstract method (does not have a body)
  public abstract void patientType();
  // Regular method
  public void patientAge() {
    System.out.println("35");
  }
}

// Subclass
class Hospital1 extends Healthcare {
  public void patientType() {

    System.out.println("The Patient Type: The patient belongs to Membership ");
  }
}

class AbstractClass1 {
  public static void main(String[] args) {
    Hospital1 Obj1 = new Hospital1();
    Obj1.patientType();
    Obj1.patientAge();
  }
}
```

Using final with inheritance

- The keyword final has three uses.
- It can be used to create the equivalent of a named constant.

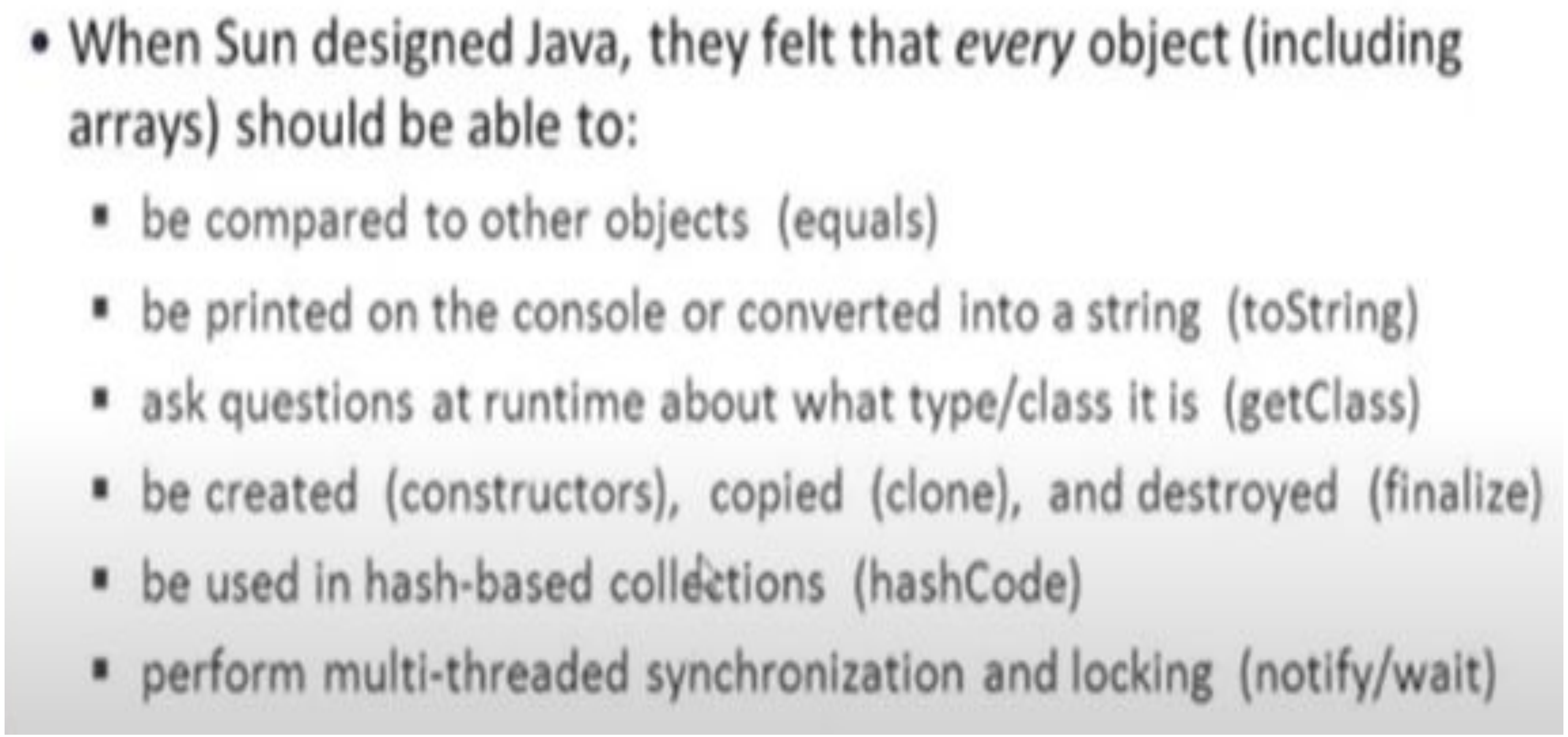
- **Using final to Prevent Overriding**

```
class A {
  final void meth() {
    System.out.println("This is a final method.");
  }
}

class B extends A {
  void meth() { // ERROR! Can't override.
    System.out.println("Illegal!");
  }
}
```

The Object Class

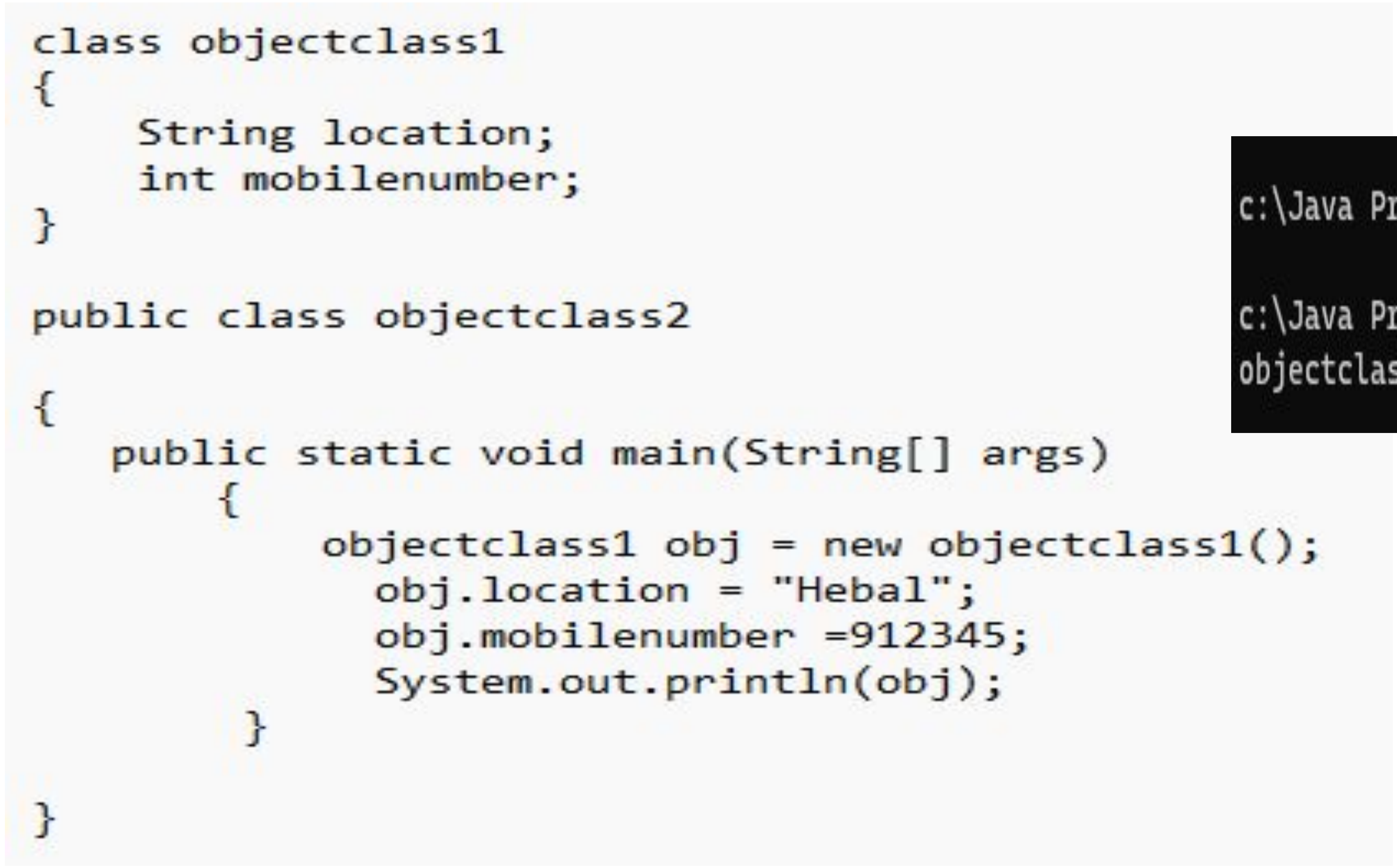
- There is one special class, Object, defined by Java. All other classes are subclasses of Object.
- **Package: java.lang**
- That is, an Object is a **superclass** of all other classes.
- Every class in java is Directly/Indirectly derived from an object class.
- This means that a reference variable of type Object can refer to an object of any other class.
- Also, since arrays are implemented as classes, a variable of type Object can also refer to an array.

Common Properties of Objects

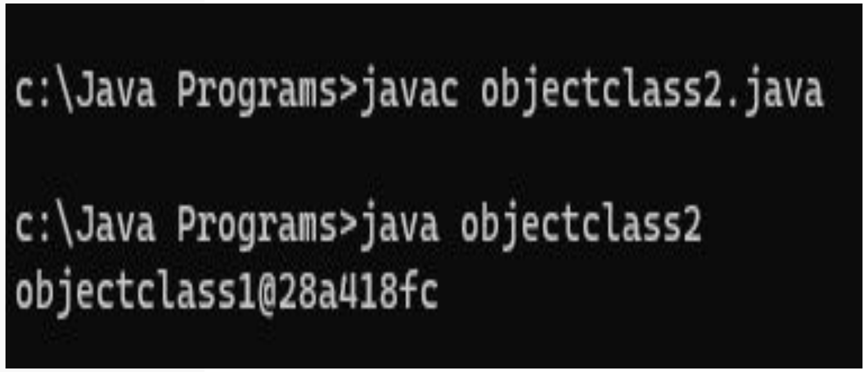


- When Sun designed Java, they felt that *every* object (including arrays) should be able to:
  - be compared to other objects (equals)
  - be printed on the console or converted into a string (toString)
  - ask questions at runtime about what type/class it is (getClass)
  - be created (constructors), copied (clone), and destroyed (finalize)
  - be used in hash-based collections (hashCode)
  - perform multi-threaded synchronization and locking (notify/wait)

Sample Pgm1

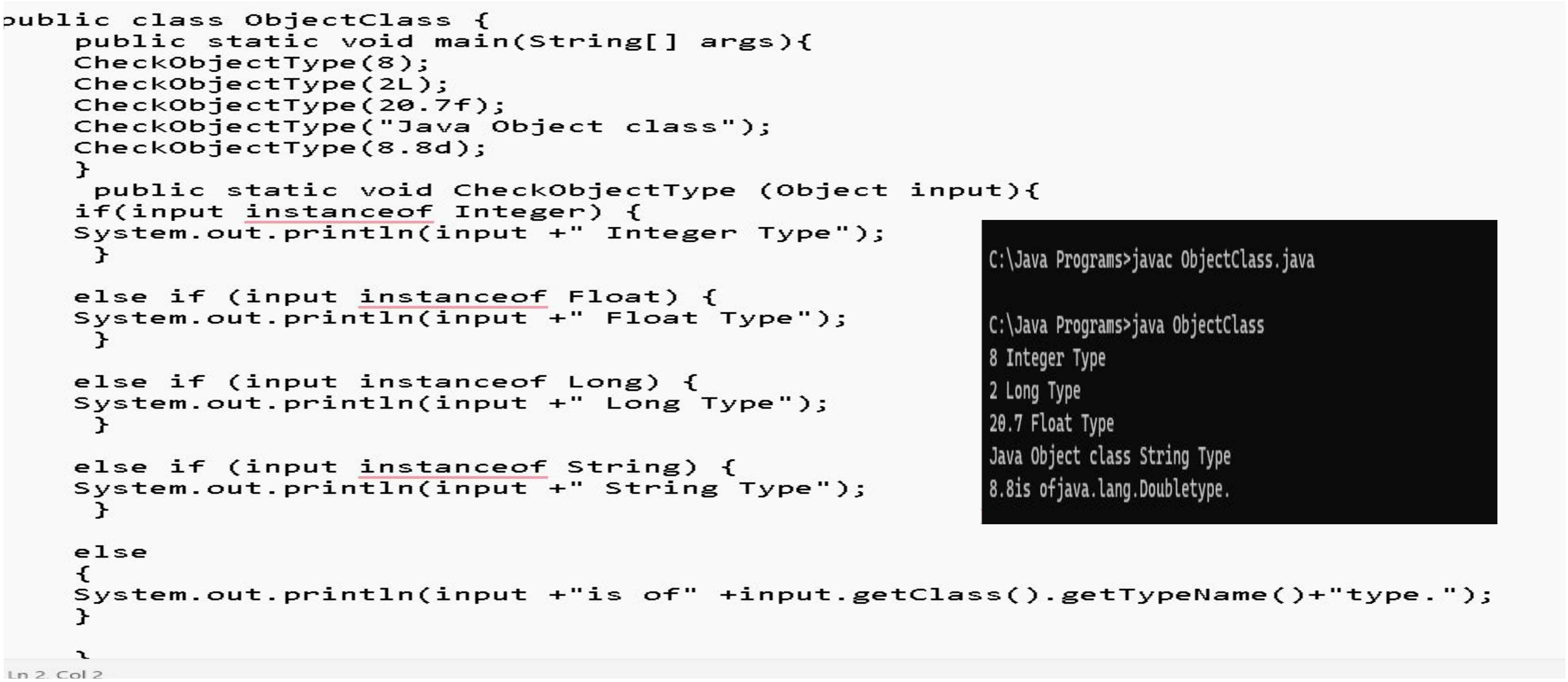


```
class objectclass1
{
    String location;
    int mobilenumber;
}

public class objectclass2

{
   public static void main(String[] args)
       {
           objectclass1 obj = new objectclass1();
             obj.location = "Hebal";
             obj.mobilenumber =912345;
             System.out.println(obj);
        }

}
```



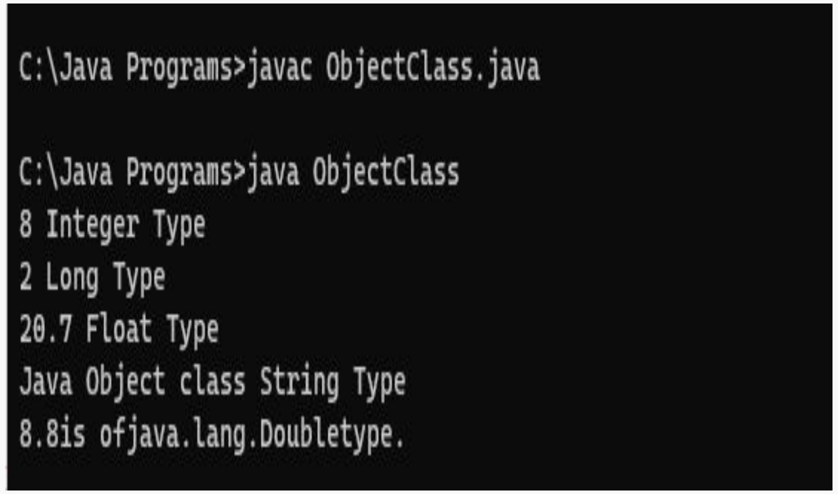
```
c:\Java Programs>javac objectclass2.java

c:\Java Programs>java objectclass2
objectclass1@28a418fc
```

Sample Pgm2



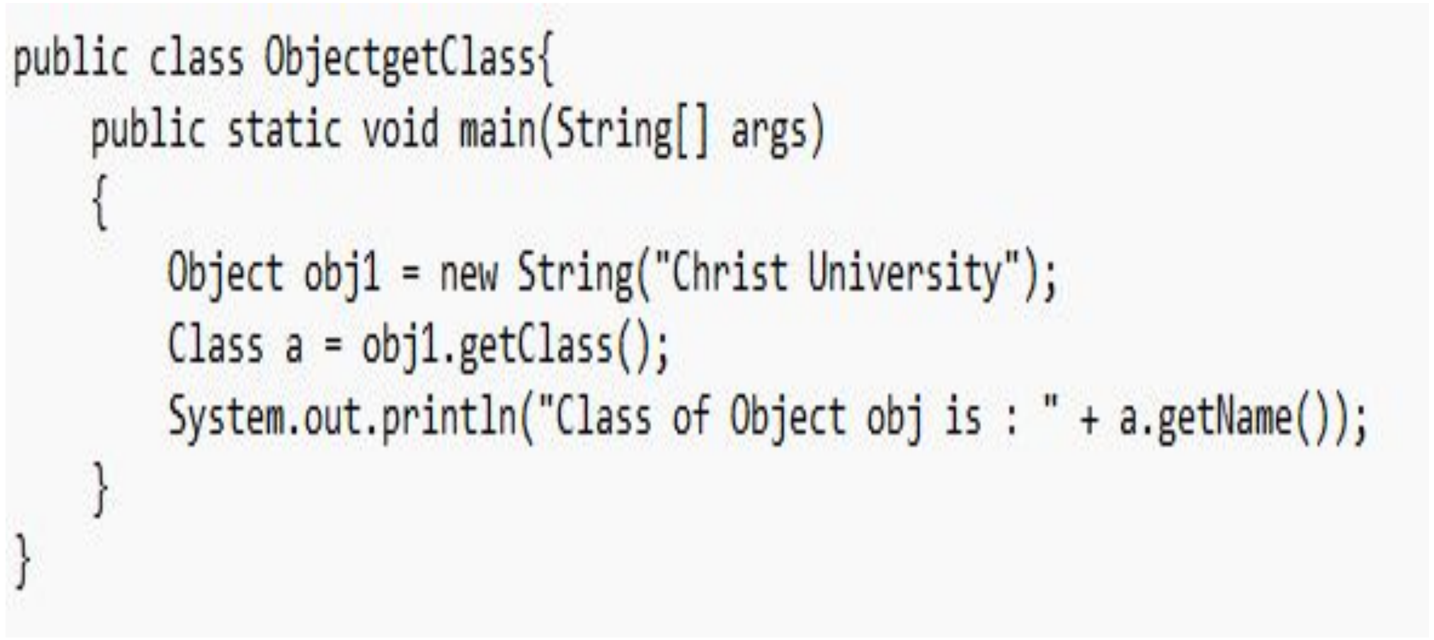
```
public class ObjectClass {
    public static void main(String[] args){
    CheckObjectType(8);
    CheckObjectType(2L);
    CheckObjectType(20.7f);
    CheckObjectType("Java Object class");
    CheckObjectType(8.8d);
    }
     public static void CheckObjectType (Object input){
    if(input instanceof Integer) {
    System.out.println(input +" Integer Type");
     }

    else if (input instanceof Float) {
    System.out.println(input +" Float Type");
     }

    else if (input instanceof Long) {
    System.out.println(input +" Long Type");
     }

    else if (input instanceof String) {
    System.out.println(input +" String Type");
     }

    else
    {
    System.out.println(input +"is of" +input.getClass().getTypeName()+"type.");
    }
```

Ln 2, Col 2

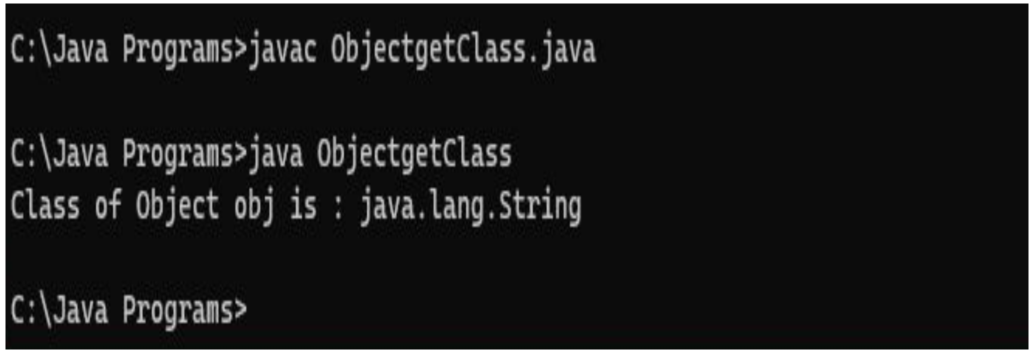


```
C:\Java Programs>javac ObjectClass.java

C:\Java Programs>java ObjectClass
8 Integer Type
2 Long Type
20.7 Float Type
Java Object class String Type
8.8is ofjava.lang.Doubletype.
```

Sample Program

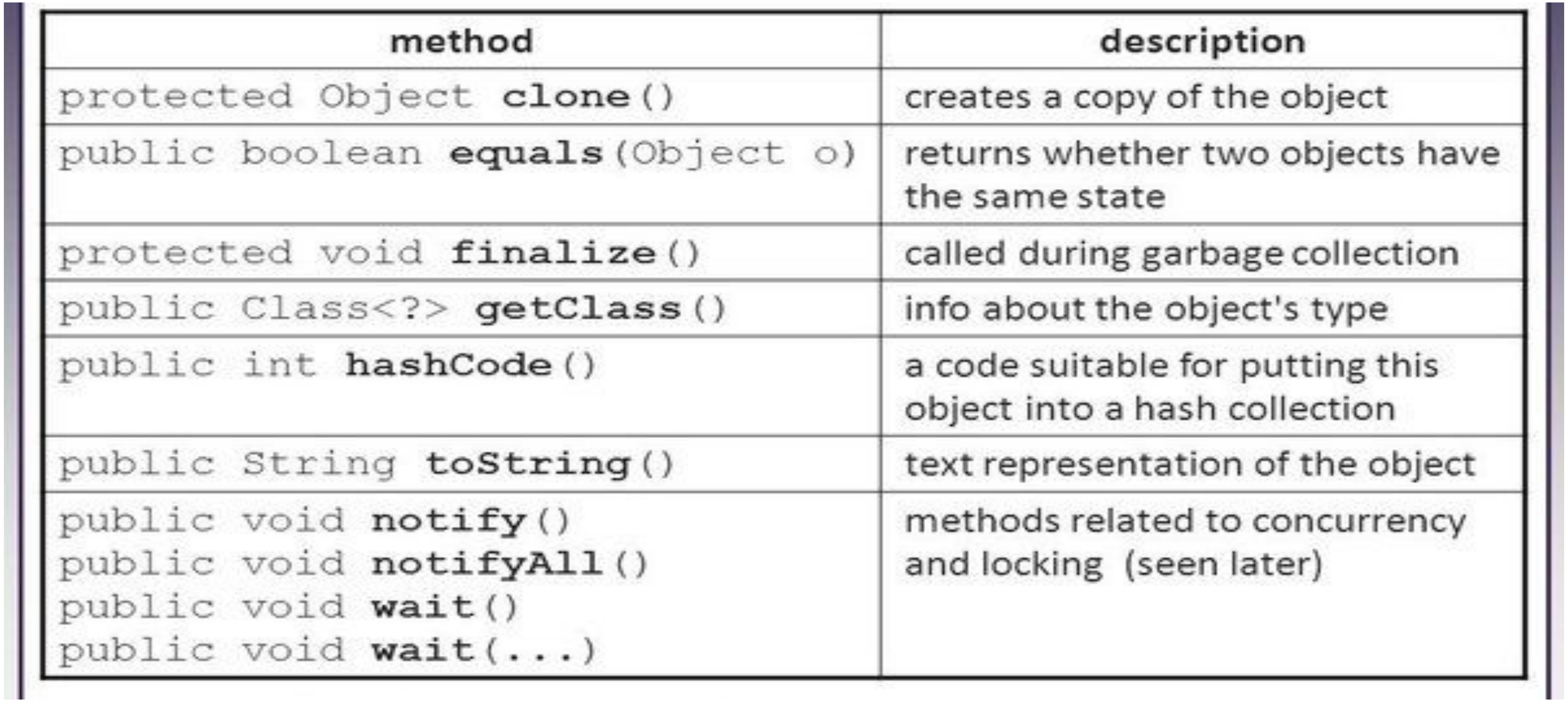


```
public class ObjectgetClass{
    public static void main(String[] args)
    {
        Object obj1 = new String("Christ University");
        Class a = obj1.getClass();
        System.out.println("Class of Object obj is : " + a.getName());
    }
}
```



```
C:\Java Programs>javac ObjectgetClass.java

C:\Java Programs>java ObjectgetClass
Class of Object obj is : java.lang.String

C:\Java Programs>
```

Object Methods



method	description
`protected Object clone()`	creates a copy of the object
`public boolean equals(Object o)`	returns whether two objects have the same state
`protected void finalize()`	called during garbage collection
`public Class<?> getClass()`	info about the object's type
`public int hashCode()`	a code suitable for putting this object into a hash collection
`public String toString()`	text representation of the object

`public void notify()`
`public void notifyAll()`
`public void wait()`
`public void wait(...)`
	methods related to concurrency and locking (seen later)