1. Make a class **Employee** with a name and salary. Make a class **Manager** inherit from Employee. Add an instance variable, named **department**, of the type String. Supply a **toString** method that prints the manager's name, department, and salary. Make a class **Executive** inherit from **Manager**. Supply appropriate **toString** methods for all classes. Provide a driver program, **EmployeeTester**, that tests these classes and methods.

```java
// Problem 1
// Driver Program
public class Employee Tester
{
    public static void main(String[] args)
    {
        Employee e1 = new Employee("Smith", 5000);
        System.out.println(e1);
        System.out.println();
        Manager m1 = new Manager("Jones", 15000, "Finance");
        System.out.println(m1);
        System.out.println();
        Manager ex1 = new Executive("Gibbs", 150000, "Finance", "CFO");
        System.out.println(ex1);
    }
}

// Class Employee
public class Employee
{
    protected String empName;
    protected int empSalary;

    public Employee(String name, int salary)
    {
        empName = name;
        empSalary = salary;
    }

    public String toString()
    {
        return "Employee name: " + empName + "\n" + "Salary: " + empSalary;
    }
}
```

**Sample Run:**

Employee name: Smith  
Salary: 5000

Employee name: Jones  
Salary: 15000  
Department: Finance

Executive name: Gibbs  
Salary: 150000  
Department: Finance  
Title: CFO
//Class Manager
public class Manager extends Employee
{
    String department;
    public Manager (String name, int salary, String dept)
    {
        super(name, salary);
        department = dept;
    }
    public String getDept()
    {
        return department;
    }
    public String toString()
    {
        return super.toString() + "\n" + "Department: " + department;
    }
}

//Class Executive
public class Executive extends Manager
{
    String title;
    public Executive (String name, int salary, String dept, String ttl)
    {
        super(name, salary, dept);
        title = ttl;
    }
    public String getDept()
    {
        return department;
    }
    public String toString()
    {
        return "Executive name: " + empName + "\n" + "Salary: " + empSalary + "\n" + "Department: " + department + "\n" + "Title: " + title;
    }
}
2. Implement a superclass **Person**. Make two classes, **Student** and **Instructor**, that inherit from **Person**. A person has a name and year of birth. A student has a major, and an instructor has a salary. Write the class declarations, the constructors, and the methods **toString** for all classes. Write a driver program to test these classes and methods.

```java
//Problem 2
public class personTest
{
    public static void main(String[] args)
    {
        Person p1 = new Person("Freshman", 1997);
        System.out.println(p1);
        System.out.println();

        Student s1 = new Student("Teekel", 1995, "Computer Science");
        System.out.println(s1);
        System.out.println();

        Instructor i1 = new Instructor("Teacher", 1980, 45000);
        System.out.println(i1);
    }
}

//Class Person

public class Person
{
    protected String perName;
    protected int birYear;

    public Person(String name, int year)
    {
        perName = name;
        birYear = year;
    }
    public String toString()
    {
        return "Person's name: " + perName + "\n" + "Birth year: " + birYear;
    }
}
```

**Sample Run:**

Person's name: Freshman
Birth year: 1997

Student name: Teekel
Birth Year: 1995
Major: Computer Science

Instructor name: Teacher
Birth Year: 1980
Salary: 45000
//Class Student

public class Student extends Person
{
    String Major;
    public Student (String name, int byear, String maj)
    {
        super(name, byear);
        Major = maj;
    }
    public String getMajor()
    {
        return Major;
    }
    public String toString()
    {
        return "Student name: " + perName + "\n" + "Birth Year: " + byYear + "\n" + "Major: " + Major;
    }
}

//Class Instructor

public class Instructor extends Person
{
    int Salary;
    public Instructor (String name, int byear, int sal)
    {
        super(name, byear);
        Salary = sal;
    }
    public int getSalary()
    {
        return Salary;
    }
    public String toString()
    {
        return "Instructor name: " + perName + "\n" + "Birth Year: " + byYear + "\n" + "Salary: " + Salary;
    }
}
3. Implement a superclass Appointment and subclasses Onetime, Daily, and Monthly. An appointment has a description (for example, “see the dentist”) and a date. Write a method occursOn (int year, in month, in day) that checks whether the appointment occurs on that date. For example, for a monthly appointment, you must check the day of the month matches. Then fill an array of Appointment objects with a mixture of appointments. Have the user enter a date and print out all appointments that occur on that date.

    public class AppointmentTest {
        public static void main(String[] args) {
            Date d1 = new Date("January", 2, 2015);
            Date d2 = new Date("January", 5, 2015);
            Date d3 = new Date("February", 2, 2015);
            Date d4 = new Date("March", 2, 2015);
            Onetime o1 = new Onetime("Doctor's visit", d2);
            Monthly m1 = new Monthly("Doctor's visit", d2);
            Daily da1 = new Daily("Doctor's visit", d2);
            if (o1.occursOn(d2))
                System.out.println("Onetime Appointment occurs on: " + d2.toString());
            if (o1.occursOn(d3))
                System.out.println("Onetime Appointment occurs on: " + d3.toString());
            if (o1.occursOn(d4))
                System.out.println("Onetime Appointment occurs on: " + d4.toString());
            if (m1.occursOn(d2))
                System.out.println("Monthly Appointment occurs on: " + d2.toString());
            if (m1.occursOn(d3))
                System.out.println("Monthly Appointment occurs on: " + d3.toString());
            if (m1.occursOn(d4))
                System.out.println("Monthly Appointment occurs on: " + d4.toString());
            if (da1.occursOn(d2))
                System.out.println("Daily Appointment occurs on: " + d2.toString());
            if (da1.occursOn(d3))
                System.out.println("Daily Appointment occurs on: " + d3.toString());
            if (da1.occursOn(d4))
                System.out.println("Daily Appointment occurs on: " + d4.toString());
        }
    }

Sample Run:

    Onetime Appointment occurs on: January 5, 2015
    Monthly Appointment occurs on: January 5, 2015
    Daily Appointment occurs on: January 5, 2015
    Daily Appointment occurs on: February 2, 2015
    Daily Appointment occurs on: March 2, 2015

5
public class Appointment {
    Date d;
    String appointment;

    public Appointment(String a, Date d1){
        d=d1;
        appointment=a;
    }
    public Date getDate(){
        return d;
    }
    public String getAppointment(){
        return appointment;
    }
    public String toString(){
        return appointment+" on "+d.toString();
    }
}

public class Date {
    private String month;
    private int day, year;

    public Date(String m, int d, int y){
        month=m;
        day=d;
        year=y;
    }

    public String getMonth(){
        return month;
    }
    public int getDay(){
        return day;
    }
    public int getYear(){
        return year;
    }
    public String toString(){
        return month+" \"+day+"\",+year;
    }
}
public class Daily extends Appointment {
    public Daily(String a, Date d1) {
        super(a, d1);
    }
    public boolean occursOn(Date d) {
        return true;
    }
}

public class Onetime extends Appointment {
    public Onetime(String a, Date d1) {
        super(a, d1);
    }
    public boolean occursOn(Date d) {
        if (super.getDate().getMonth().equalsIgnoreCase(d.getMonth())
            && super.getDate().getDay() == d.getDay()) {
            if (super.getDate().getYear() == d.getYear())
                return true;
        }
        return false;
    }
}

public class Monthly extends Appointment {
    public Monthly(String a, Date d1) {
        super(a, d1);
    }
    public boolean occursOn(Date d) {
        if (super.getDate().getDay() == d.getDay() && super.getDate().getYear() == d.getYear())
            return true;
        return false;
    }
}
4. Design and implement a program that creates an exception class called
InvalidDocumentCodeException, designed to be thrown when an improper designation for
document is encountered during processing. Suppose in a particular business all documents are
given two-character designation starting with either U, C, or P, standing for unclassified,
confidential, or proprietary. If a document designation is encountered that does not fit that
description, the exception is thrown. Create a driver program to test the exception, allowing it
to terminate the program.

//Problem 4
//Driver Program
import java.util.Scanner;
public class DocDesignation
{
    public static void main(String[] args)
    {
        char des;
        int district, valid = 0, banned = 0;
        String code;
        Scanner scan = new Scanner(System.in);
        InvalidDocumentCodeException e;
        System.out.print("Enter document name/designation: ");
        code = scan.nextLine();
        des = code.charAt(code.length()-1);
        try
        {
            if (des == 'U')
                System.out.println("The document is: Unclassified");
            else if (des == 'C')
                System.out.println("The document is: Classified");
            else if (des == 'P')
                System.out.println("The document is: Proprietary");
            else
                throw new InvalidDocumentCodeException("");
        }
        catch (InvalidDocumentCodeException exception)
        {
            System.out.println("Exception thrown, Code detected: "+des);
        }
    }
}

// Class InvalidDocumentCodeException
public class InvalidDocumentCodeException extends Exception {
    public InvalidDocumentCodeException(String message) {
        super(message);
    }
}

Sample Run:
Enter document name/designation: 123C
The document is: Classified
Enter document name/designation: ABCU
The document is: Unclassified
5. Write a class **BankAccount**, with two subclasses, **Savings** and **Checking** that inherit from it. Savings account holders earn an interest of 1.5% and maintain a minimum of $1000 to avoid monthly service charges of $15.00. Checking account holders are required to maintain $500 minimum to avoid the $15 service charge. Both accounts are linked and can use an ATM machine for deposits and withdrawals. A service charge of $4.00 is assessed for using non-member ATM machines. Start with an abstract **BankAccount** class and derive the two subclasses: Saving and Checking. Create several account holder objects with different starting balances and different set of transactions and prepare monthly statements for each. Sample report includes:

<table>
<thead>
<tr>
<th>Account Name: ...</th>
<th>Account Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account number: ...</td>
<td>Account Type: ...</td>
</tr>
<tr>
<td>Starting balance: ...</td>
<td>Ending Balance: ...</td>
</tr>
<tr>
<td>Number of transactions: ...</td>
<td>Service Charge: ...</td>
</tr>
<tr>
<td>Total Debit: ....</td>
<td>Total Credit: ....</td>
</tr>
<tr>
<td>Interest Earned: ....</td>
<td>Service Charge: ...</td>
</tr>
</tbody>
</table>
Problem 5
public class BankAccountsTester
{
    public static void main(String[] args)
    {
        Savings s1 = new Savings("Genesis",123456789,5000);
        Checking c1 = new Checking("Melissa",1234567891, 1000);
        System.out.println(s1);
        System.out.println("\n"+c1);
        c1.makeDeposit(500,"NM");
        c1.makeWithdrawal(1000,"M");
        System.out.println("\nThe balance in Melissa's account is:
"+c1.getBalance());
        System.out.println("now it is: "+c1.getBalance());
        s1.makeWithdrawal(1000,"NM");
        System.out.println("\nThe balance in Genesis' account is:
"+s1.getBalance());
        Savings s2= new Savings("Ed",551625369,1000);
        s2.makeDeposit(500,"NM");
        Checking c2=new Checking("Tony",156785420,500);
        System.out.println("\n"+s1.getMonthlyStatement());
        System.out.println("\n"+c1.getMonthlyStatement());
        System.out.println("\n"+s2.getMonthlyStatement());
        System.out.println("\n"+c2.getMonthlyStatement());
    }
}

public abstract class BankAccount
{
    protected String acctName;
    protected long acctNumber;
    protected double balance;

    public BankAccount(String n, long num, double b)
    {
        acctName=n;
        acctNumber = num;
        balance=b;
    }

    public String toString()
    {
        return "Account name: "+acctName+"\n"+"Account number: "+
                acctNumber+"\n"+"Balance: "+balance;
    }

Sample Run:
Account name: Genesis
Account number: 123456789
Balance: 5000.0

Account name: Melissa
Account number: 1234567891
Balance: 1000.0
The balance in Melissa's account is: 496.0
now it is: 496.0
The balance in Genesis' account is: 3996.0

Account name: Genesis
Account Number: 123456789
Account type: Savings
Starting Balance: 5000.0
Number of Transactions: 1
Interest earned: 59.94
Ending balance: 4055.94
Service Charge: 0.0

Account name: Melissa
Account Number: 1234567891
Account type: Checking
Starting Balance: 0.0
Number of Transactions: 2
Ending balance: 481.0
Service Charge: 15.0

Account name: Ed
Account Number: 551625369
Account type: Savings
Starting Balance: 1000.0
Number of Transactions: 1
Interest earned: 22.5
Ending balance: 1522.5
Service Charge: 0.0

Account name: Tony
Account Number: 156785420
Account type: Checking
Starting Balance: 0.0
Number of Transactions: 0
Ending balance: 500.0
Service Charge: 15.0
public class Savings extends BankAccount
{
    private double interest, serviceFee, atmFee, startingBal, interestEarned;
    private int totalTrans;

    public Savings(String n, long num, double b)
    {
        super(n, num, b);
        interest = 0.015;
        serviceFee = 0;
        atmFee = 0;
        startingBal = b;
        totalTrans = 0;
        interestEarned = 0;
    }
    public double getBalance()
    {
        return balance;
    }

    public void makeDeposit(double d, String atmType)
    {
        totalTrans++;
        if (atmType.equals("NM"))
        {
            atmFee = 4;
            balance = balance + d - atmFee;
        } else
        balance = balance + d;
    }

    public void makeWithdrawal(double w, String atmType)
    {
        totalTrans++;
        if (atmType.equals("NM"))
        {
            atmFee = 4;
            balance = balance - w - atmFee;
        } else
        balance = balance - w;
    }
    public String getMonthlyStatement()
    {
        if (balance < 1000)
{  
  serviceFee=15.00;  
  interestEarned=(balance-serviceFee)*interest;  
  balance += interestEarned;  
}  
else if(balance>=1000)  
{  
  interestEarned= balance*interest;  
  balance += interestEarned;  
}  
return "Account name:"+acctName+"\nAccount Number:"+acctNumber+"\nAccount type: Savings\nStarting Balance:"+startingBal  
+"\nNumber of Transactions:"+totalTrans+"\nInterest earned:"+interestEarned+"\nEnding balance:"+balance+  
+"\nService Charge:"+serviceFee;  
}

public class Checking extends BankAccount  
{  
  private double serviceFee, atmFee, startingBal;  
  private int totalTrans;  
  public Checking(String n, long num, double b)  
  {  
    super(n,num,b);  
    serviceFee = 15.00;  
    atmFee = 4.00;  
  }  
  public double getBalance()  
  {  
    return balance;  
  }  
  public void makeDeposit(double d, String atmType)  
  {  
    totalTrans++;  
    if (atmType.equals("NM"))  
    {  
      balance = balance+d-atmFee;  
    }  
    else  
    balance = balance+d;  
  }  
  public void makeWithdrawal(double w, String atmType)  
  {  
    totalTrans++;  
  }
if (atmType.equals("NM"))
{
    balance = balance-w-atmFee;
}
else
    balance = balance-w;
}

public String getMonthlyStatement()
{
    if(balance<500)
    {
        serviceFee=15.00;
        balance -= serviceFee;
    }

    return "Account name: " + acctName + "\nAccount Number: " + acctNumber + "\nAccount type: Checking\nStarting Balance: " + startingBal + "\nNumber of Transactions: " + totalTrans + "\nEnding balance: " + balance + "\nService Charge: " + serviceFee;
}