Lab4

L1 slot

V.Manikanta Bhuvanesh

19BCD7088

1.a.

Create an abstract class named Book. Include a String field for the book'stitle and a double field for the book's price. Within the class, include aconstructor that requires the book title, and add two get methods—one that returns the title and one that returns the price. Include an abstract method named setPrice(). Create two child classes of Book: Fiction and NonFiction. Each must include a setPrice() method that sets the price for all Fiction Books to $24.99 and for all NonFiction Books to $37.99. Write a constructor for each subclass, and include a call to setPrice() within each. Write an application demonstrating that you can create both a Fiction and a NonFiction Book, and display their fields. Save the files as Book.java, Fiction.java, NonFiction.java, and UseBook.java.

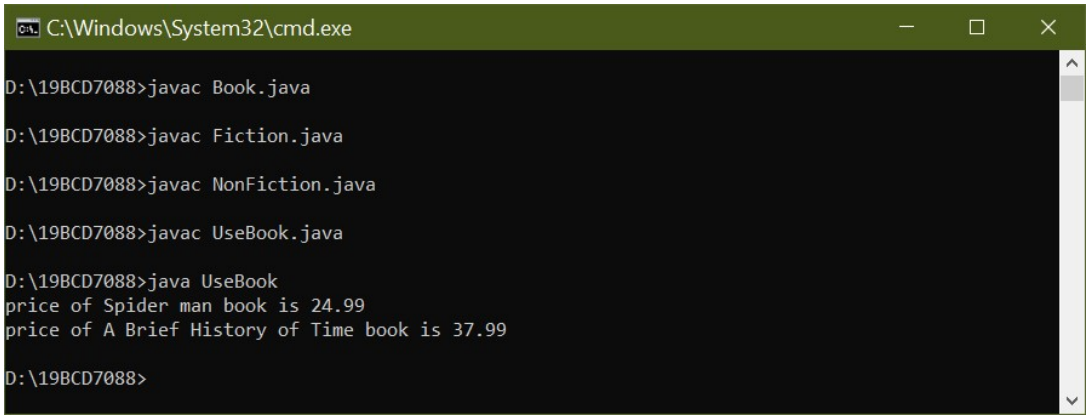
```
abstract class Book {
   String t;
   double p;

   Book(String title ){
      t=title;
   }

   String gettitle(){
      return t;
   }

   double getPrice(){
      return p;
   }

   abstract void setPrice();
}
```

```
class Fiction extends Book{

   Fiction(String title) {
      super(title);
      setPrice();
```

```
    }

    void setPrice(){
        super.p=24.99;
    }
}
```

```
class NonFiction extends Book{

    NonFiction(String title) {
        super(title);
        setPrice();
    }

    void setPrice(){
        super.p=37.99;
    }
}
```

```
public class UseBook {
    public static void main(String[] args){
        Book b1,b2;
        b1=new Fiction("Spider man");
        System.out.println("price of " + b1.gettitle() + " book is " + b1.getPrice());
        b2=new NonFiction("A Brief History of Time");
        System.out.println("price of " + b2.gettitle() + " book is " + b2.getPrice());
    }
}
```



1.b.

Write an application named BookArray in which you create an array that holds 10 Books, some Fiction and some NonFiction. Using a for loop, display details about all 10 books. Save the file as BookArray.java.

```
public class BookArray{
	public static void main(String[] args) {
		Book b[] = new Book[10];
		b[0] = new Fiction("The Paying Guests");
		b[1] = new Fiction("Nineteen Eighty-Four");
		b[2] = new Fiction("A Tale of Two Cities");
		b[3] = new Fiction("Jane Eyre");
		b[4] = new Fiction("Town Like Alice");
		b[5] = new NonFiction("A Brief History of Time");
		b[6] = new NonFiction(" I Know Why the Caged Bird Sings");
		b[7] = new NonFiction("In Cold Blood");
		b[8] = new NonFiction("Bury My Heart at Wounded Knee");
		b[9] = new NonFiction("Hiroshima");
		for(int i=0;i<10;i++){
			System.out.println("price of " + b[i].gettitle() + " book is " + b[i].getPrice());
		}
	}
}
```

```
C:\Windows\System32\cmd.exe
D:\19BCD7088>javac BookArray.java

D:\19BCD7088>java BookArray
price of The Paying Guests book is 24.99
price of Nineteen Eighty-Four book is 24.99
price of A Tale of Two Cities book is 24.99
price of Jane Eyre book is 24.99
price of Town Like Alice book is 24.99
price of A Brief History of Time book is 37.99
price of  I Know Why the Caged Bird Sings book is 37.99
price of In Cold Blood book is 37.99
price of Bury My Heart at Wounded Knee book is 37.99
price of Hiroshima book is 37.99

D:\19BCD7088>
```

2.a.

The Talk-A-Lot Cell Phone Company provides phone services for its customers. Create an abstract class named PhoneCall that includes a String field for a phone number and a double field for the price of the call. Also include a constructor that requires a phone number parameter and that sets the price to 0.0. Include a set method for the price. Also include three abstract get methods—one that returns the phone number, another that returns the price of the call, and a third that displays information about the call. Create two child classes of PhoneCall: IncomingPhoneCall and OutgoingPhoneCall. The IncomingPhoneCall constructor passes its phone number parameter to its parent's constructor and sets the price of the call to 0.02. The method that displays the phone call information displays the phone number, the rate, and the price of the call (which is the same as the rate). The OutgoingPhoneCall class includes an additional field that holds the time of the call in minutes. The constructor requires both a phone number and the time. The price is 0.04 per minute, and the display method shows the details of the call, including the phone number, the rate per minute, the number of minutes, and the total price. Write an application that demonstrates you can instantiate and display both IncomingPhoneCall and OutgoingPhoneCall objects. Save the files as PhoneCall.java, IncomingPhoneCall.java, OutgoingPhoneCall.java, and DemoPhoneCalls.java.

```
abstract class PhoneCall

{

   String phn;

   double price;

    PhoneCall(String phn)

   {

      this.phn =  phn;

      this.price = 0.0;

   }

   abstract String getPhoneNumber();

   abstract double getPrice();
```

```
    abstract void getInf();
    abstract void setPrice();

}
```

```
class IncomingPhoneCall extends PhoneCall {

    double r=0.02;
     IncomingPhoneCall(String phoneNumber){
        super(phoneNumber);
        setPrice();
    }
    void setPrice() {
        super.price = 0.02;
    }
     void getInf(){
        System.out.println("Incoming phone call for "+getPhoneNumber()+",Price for a call is
$"+getPrice());
    }
      String getPhoneNumber()
      {
         return super.phn;
      }
      double getPrice()
      {
         return super.price;
      }
}
```

```
class OutgoingPhoneCall extends PhoneCall {

    double r = 0.04;
    int minutes;
```

```
 OutgoingPhoneCall(String phoneNumber, int minutes){
    super(phoneNumber);
    this.minutes = minutes;
    setPrice();
  }
  void setPrice() {
    super.price = 0.04;
  }
  void getInf() {
    System.out.println("Outgoing phone call for " + getPhoneNumber() + " "+ r + " per minute at " +
minutes + " minutes is $" + price*minutes);
  }
 public String getPhoneNumber()
 {
    return super.phn;
 }
 public double getPrice()
 {
    return super.price;
 }

}
```

```
public class DemoPhoneCalls {
public static void main(String [] args) {

  IncomingPhoneCall in=new IncomingPhoneCall("9014914993");
  OutgoingPhoneCall out=new OutgoingPhoneCall("9456552237",40);
  in.getInf();
  out.getInf();
 }
}
```

```
C:\Windows\System32\cmd.exe

D:\19BCD7088>javac PhoneCall.java

D:\19BCD7088>javac IncomingPhoneCall.java

D:\19BCD7088>javac OutgoingPhoneCall.java

D:\19BCD7088>javac DemoPhoneCalls.java

D:\19BCD7088>java DemoPhoneCalls
Incoming phone call for 9014914993,Price for a call is $0.02
Outgoing phone call for 9456552237 0.04 per minute at 40 minutes is $1.6

D:\19BCD7088>
```

2.b.

Write an application in which you assign data to a mix of eight IncomingPhoneCall and OutgoingPhoneCall objects into an array. Use a for loop to display the data. Save the file as PhoneCallArray.java.

```
public class PhoneCallArray{
        public static void main(String[] args) {
                IncomingPhoneCall in[]=new IncomingPhoneCall[3];
                OutgoingPhoneCall out[]=new OutgoingPhoneCall[5];
                in[0]=new IncomingPhoneCall("9856412345");
                in[1]=new IncomingPhoneCall("94851265126");
                in[2]=new IncomingPhoneCall("78843204566");
                out[0]= new OutgoingPhoneCall("8456131361",50);
                out[1]= new OutgoingPhoneCall("9456123789",60);
                out[2]= new OutgoingPhoneCall("7541237896",70);
                out[3]= new OutgoingPhoneCall("7148529632",80);
                out[4]= new OutgoingPhoneCall("8794561235",90);
                for(int i=0;i<3;i++){
                        in[i].getInf();
                }
                for(int j=0;j<5;j++){
                        out[j].getInf();
                }
        }
```

}

```
C:\Windows\System32\cmd.exe

D:\19BCD7088>java PhoneCallArray
Incoming phone call for 9856412345,Price for a call is $0.02
Incoming phone call for 94851265126,Price for a call is $0.02
Incoming phone call for 78843204566,Price for a call is $0.02
Outgoing phone call for 8456131361 0.04 per minute at 50 minutes is $2.0
Outgoing phone call for 9456123789 0.04 per minute at 60 minutes is $2.4
Outgoing phone call for 7541237896 0.04 per minute at 70 minutes is $2.8000000000000003
Outgoing phone call for 7148529632 0.04 per minute at 80 minutes is $3.2
Outgoing phone call for 8794561235 0.04 per minute at 90 minutes is $3.6

D:\19BCD7088>
```

3.a.

Create an interface named Turner, with a single method named turn(). Create a class named Leaf that implements turn() to display Changing colors. Create a class named Page that implements turn() to display Going to the next page. Create a class named Pancake that implements turn() to display Flipping. Write an application named DemoTurners that creates one object of each of these class types and demonstrates the turn() method for each class. Save the files as Turner.java, Leaf.java, Page.java, Pancake.java, and DemoTurners.java.

```
interface Turner {
   public void turn();
}
```

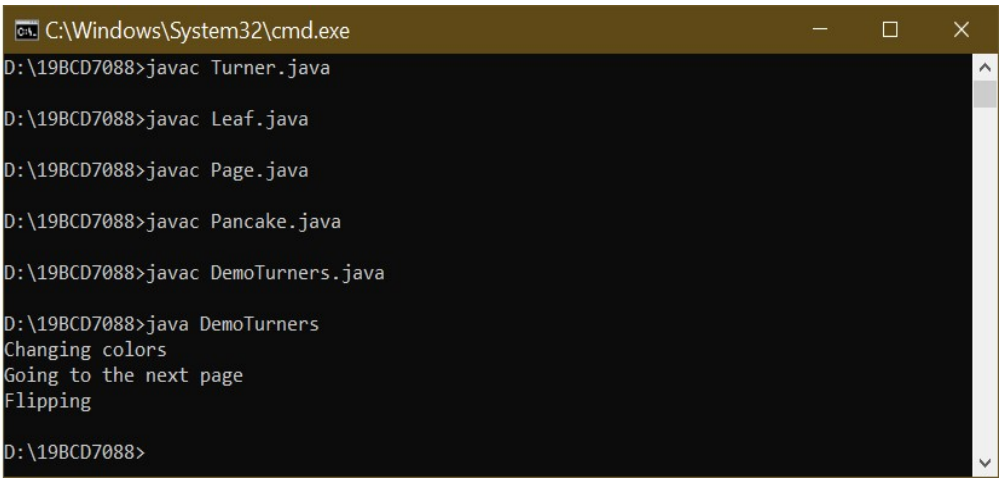
```
class Leaf implements Turner{
   public void turn() {
      System.out.println("Changing colors");
   }
}
```

```
class Page implements Turner{
   public void turn() {
      System.out.println("Going to the next page");
   }
}
```

```
class Pancake implements Turner{
   public void turn() {
```

```
        System.out.println("Flipping");
    }
}
```

```
public class DemoTurners {
    public static void main(String[] args){
        Leaf l = new Leaf();
        Page p=new Page();
        Pancake c=new Pancake();
        l.turn();
        p.turn();
        c.turn();
    }
}
```



3.b.

Think of two more objects that use turn(), create classes for them, and then add objects to the DemoTurners application, renaming it DemoTurners2. java. Save the files, using the names of new objects that use turn().

```
interface Turner {
    public void turn();
}
```

```
class Leaf implements Turner{
    public void turn() {
```

```
        System.out.println("Changing colors");
    }
}
```

```
class Page implements Turner{
    public void turn() {
        System.out.println("Going to the next page");
    }
}
```

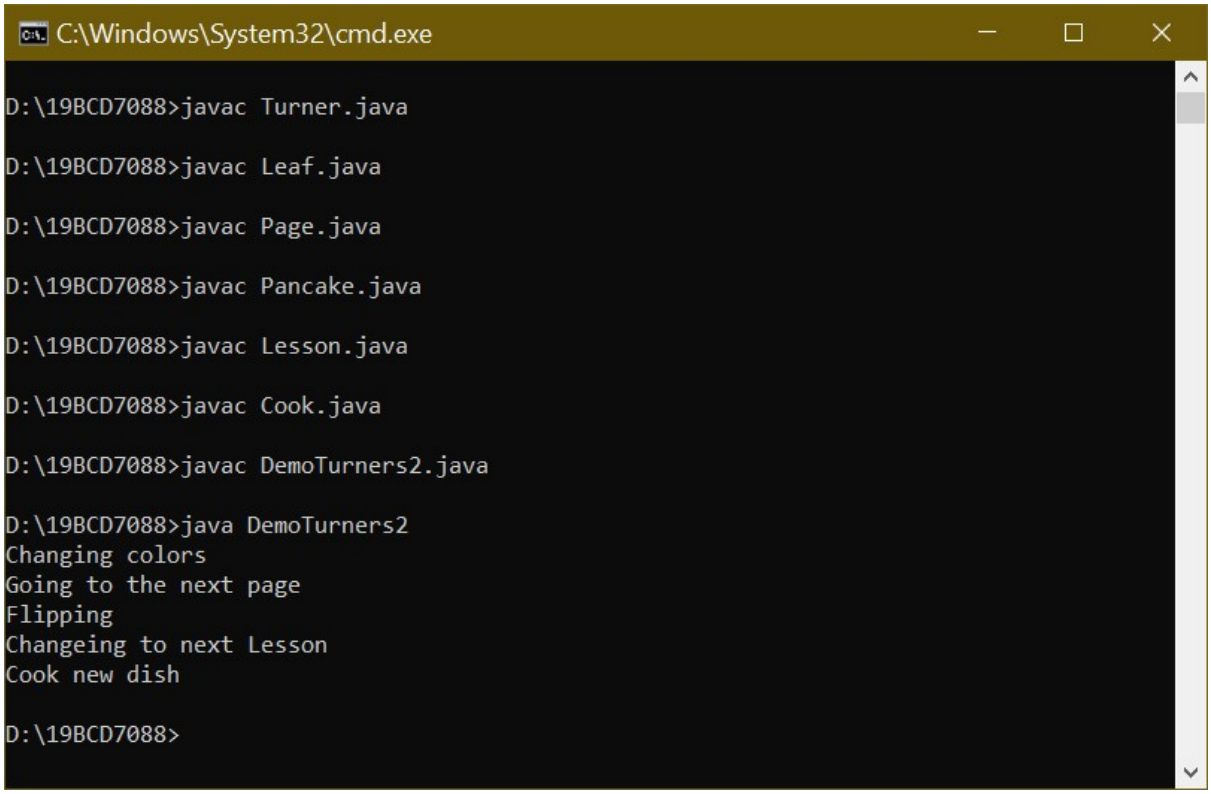
```
class Pancake implements Turner{
    public void turn() {
        System.out.println("Flipping");
    }
}
```

```
class Lesson implements Turner{
    public void turn() {
        System.out.println("Changeing to next Lesson");
    }
}
```

```
class Cook implements Turner{
    public void turn() {
        System.out.println("Cook new dish");
    }
}
```

```
public class DemoTurners2 {
    public static void main(String[] args){
        Leaf l = new Leaf();
        Page p=new Page();
        Pancake c=new Pancake();
        Lesson le = new Lesson();
        Cook co = new Cook();
```

```
        l.turn();
        p.turn();
        c.turn();
        le.turn();
        co.turn();
    }
}
```



3.c.

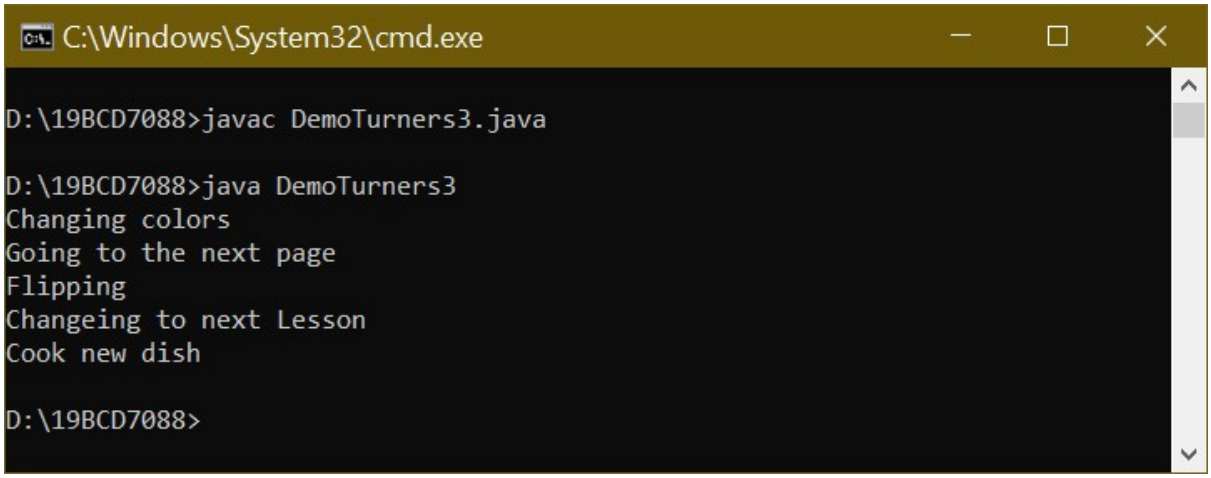
Apply Dynamic method dispatch to show the power of it and name the class as DemoTurners3.

```
interface Turner {
    public void turn();
}
```

```
class Leaf implements Turner{
    public void turn() {
        System.out.println("Changing colors");
    }
```

```
}
class Page implements Turner{
   public void turn() {
      System.out.println("Going to the next page");
   }
}
class Pancake implements Turner{
   public void turn() {
      System.out.println("Flipping");
   }
}
class Lesson implements Turner{
   public void turn() {
      System.out.println("Changeing to next Lesson");
   }
}
class Cook implements Turner{
   public void turn() {
      System.out.println("Cook new dish");
   }
}
public class DemoTurners3 {
   public static void main(String[] args){
      Turner t;
      t = new Leaf();
      t.turn();
      t=new Page();
      t.turn();
      t=new Pancake();
      t.turn();
```

```
        t= new Lesson();
        t.turn();
        t = new Cook();
        t.turn();
    }
}
```



4.a.

Create an abstract class called GeometricFigure. Each figure includes a height, a width, a figure type, and an area. Include an abstract method to determine the area of the figure. Create two subclasses called Square and Triangle. Create an application that demonstrates creating objects of both subclasses, and store them in an array. Save the files as GeometricFigure.java, Square.java, Triangle.java, and UseGeometric.java.

```
abstract class GeometricFigure {
    int height, width;
    String figureType;
    int area;
    abstract void Area(int h, int w);
}
```

```
class Square extends GeometricFigure{
    Square(int a, int b){
        super.height=a;
        super.width=b;
```

```
        Area(a,b);
    }
    void Area(int h, int w) {
      super.area=(h*w);
    }
}
```

```
class Triangle extends GeometricFigure{

    Triangle(int a, int b){
        super.height=a;
        super.width=b;
        Area(a,b);
    }
    void Area(int h, int w) {
        area=(h*w)/2;
    }

}
```

```
public class UseGeometric {
    public static void main(String[] args){
        GeometricFigure f[]=new GeometricFigure[2];
        Square s=new Square(20,20);
        Triangle t=new Triangle(10,20);
        f[0]=s;
        f[1]=t;
    }

}
```

```
C:\Windows\System32\cmd.exe

D:\19BCD7088>javac GeometricFigure.java

D:\19BCD7088>javac Square.java

D:\19BCD7088>javac Triangle.java

D:\19BCD7088>javac UseGeometric.java

D:\19BCD7088>java UseGeometric

D:\19BCD7088>
```

4.b.

Modify 4.a., adding an interface called SidedObject that contains a method called displaySides(); this method displays the number of sides the object possesses. Modify the GeometricFigure subclasses to include the use of the interface to display the number of sides of the figure. Create an application that demonstrates the use of both subclasses. Save the files as GeometricFigure2.java, Square2.java, Triangle2.java, SidedObject.java, and UseGeometric2.java.

```
abstract class GeometricFigure {
   int height, width;
   String figureType;
   int area,sides;
   abstract void Area(int h, int w);
}
```

```
interface SidedObject{
        public void display();
}
```

```
class Square2 extends GeometricFigure implements SidedObject{
   Square2(int a, int b){
      super.height=a;
      super.width=b;
      super.sides=4;
      Area(a,b);
   }
   void Area(int h, int w) {
```

```
    super.area=(h*w);
  }
  public void display(){
        System.out.println("Area of the figure is " + super.area + " and number of sides are " +
super.sides);
  }
}
```

```
class Triangle2 extends GeometricFigure implements SidedObject{


  Triangle2(int a, int b){
    super.height=a;
    super.width=b;
    super.sides=3;
    Area(a,b);
  }
  void Area(int h, int w) {
    super.area=(h*w)/2;
  }
  public void display(){
        System.out.println("Area of the figure is " + super.area + " and number of sides are " +
super.sides);
  }
}
```

```
public class UseGeometric2 {
  public static void main(String[] args){
    Square2 s=new Square2(20,20);
    Triangle2 t=new Triangle2(10,20);
    s.display();
    t.display();
  }
```

}

```
C:\Windows\System32\cmd.exe
D:\19BCD7088>javac GeometricFigure2.java

D:\19BCD7088>javac SidedObject.java

D:\19BCD7088>javac Square2.java

D:\19BCD7088>javac Triangle2.java

D:\19BCD7088>javac UseGeometric2.java

D:\19BCD7088>java UseGeometric2
Area of the figure is 400 and number of sides are 4
Area of the figure is 100 and number of sides are 3

D:\19BCD7088>
```

5.

Sanchez Construction Loan Co. makes loans of up to $100,000 for construction projects. There are two categories of Loans—those to businesses and those to individual applicants.

Write an application that tracks all new construction loans. The application also must calculate the total amount owed at the due date (original loan amount + loan fee). The application should include the following classes:

•• Loan—A public abstract class that implements the LoanConstants interface. A Loan includes a loan number, customer last name, amount of loan, interest rate, and term. The constructor requires data for each of the fields except interest rate. Do not allow loan amounts greater than $100,000. Force any loan term that is not one of the three defined in the LoanConstants class to a short-term, 1-year loan. Create a toString() method that displays all the loan data.

•• LoanConstants—A public interface class. LoanConstants includes constant values for short-term (1 year), medium-term (3 years), and long-term (5 years) loans. It also contains constants for the company name and the maximum loan amount.

•• BusinessLoan—A public class that extends Loan. The BusinessLoan constructor sets the interest rate to 1% more than the current prime interest rate.

•• PersonalLoan—A public class that extends Loan. The PersonalLoan constructor sets the interest rate to 2% more than the current prime interest rate.

•• CreateLoans—An application that creates an array of five Loans. Prompt the user for the current prime interest rate. Then, in a loop, prompt the user for a loan type and all relevant information for that loan. Store the created Loan objects in the array. When data entry is complete, display all the loans.

Save the files as Loan.java, LoanConstants.java, BusinessLoan.java, PersonalLoan.java, and CreateLoans.java.

```
interface LoanConstants {
   public int st = 1;
   public int mt = 3;
   public int lt = 5;
   public String cn = "Sanchez Construction Loan Co.";
   public double max = 100000;
}
```

```
abstract class Loan implements LoanConstants {
   String loanNum;
   String lastName;
   double loanAmt;
   double interestRate;
   int term;
   Loan(String loanNum, String lastName, double loanAmt, int term) {
      this.loanNum = loanNum;
      this.lastName = lastName;
      if (loanAmt > max) {
            System.out.println("Loan amount value is more than $100,000");
         }
      else {
            this.loanAmt = loanAmt;
         }
      if(term==st||term==mt||term==lt){
         this.term=term;
      }
```

```
        else{
            this.term=1;
        }


    }
    public String toString() {
        double n =this.loanAmt+(this.loanAmt * (this.interestRate/100));
        return this.lastName + "'s loan number is " + this.loanNum + " his loan amount is " +
this.loanAmt + " with intrest rate of " + this.interestRate +" and total due is " + n + " in term " +
this.term;
    }
}
```

```
class BusinessLoan extends Loan {
    BusinessLoan(String loanNum, String lastName, double loanAmt, int term, double primeIntRate) {
        super(loanNum, lastName, loanAmt, term);
        super.interestRate = 0.01 +primeIntRate;
    }
}
```

```
class PersonalLoan extends Loan {
    PersonalLoan(String loanNum, String lastName, double loanAmt, int term, double primeIntRate) {
        super(loanNum, lastName, loanAmt, term);
        super.interestRate = 0.02 + primeIntRate ;
    }
}
```

```
import java.util.Scanner;
public class CreateLoans{
        public static void main(String[] args) {
                Scanner sc = new Scanner(System.in);
                Loan[] l = new Loan[5];
                String ch,ln,lon,p;
                double amt,ir;
```

```
int term;
for(int i=0;i<5;i++){
   System.out.println("Enter b for BusinessLoan & p for personal loan");
   if(i!=0){
      sc.nextLine();
   }
            ch=sc.nextLine();
            System.out.println("Enter lastName");
            ln=sc.nextLine();
            System.out.println("Enter loan number");
            lon=sc.nextLine();
            System.out.println("Enter amount");
            amt=sc.nextDouble();
            System.out.println("prime interest rate");
            ir=sc.nextDouble();
            System.out.println("Enter term number");
            term=sc.nextInt();
            if(ch.equals("b")){
                        l[i]=new BusinessLoan(lon,ln,amt,term,ir);
            }
            else if(ch.equals("p")){
                        l[i]=new PersonalLoan(lon,ln,amt,term,ir);
            }
            else{
                        System.out.println("Invalid loan type");
            }
}
for(int j = 0;j<5;j++){
            p=l[j].toString();
            System.out.println(p);
}
```

```
        }
}
```

C:\Windows\System32\cmd.exe

```
D:\19BCD7088>javac LoanConstants.java

D:\19BCD7088>javac Loan.java

D:\19BCD7088>javac BusinessLoan.java

D:\19BCD7088>javac PersonalLoan.java

D:\19BCD7088>javac CreateLoans.java

D:\19BCD7088>java CreateLoans
Enter b for BusinessLoan & p for personal loan
b
Enter lastName
Valiveti
Enter loan number
12bvgf2342
Enter amount
90000
prime interest rate
1
Enter term number
1
Enter b for BusinessLoan & p for personal loan
p
Enter lastName
Thota
Enter loan number
13vbjd5643
Enter amount
70000
prime interest rate
2
Enter term number
2
Enter b for BusinessLoan & p for personal loan
p
Enter lastName
Guntur
Enter loan number
14ncjd7865
Enter amount
80000
prime interest rate
2
Enter term number
5
Enter b for BusinessLoan & p for personal loan
b
Enter lastName
Sikakoli
Enter loan number
15vxgs5467
Enter amount
50000
prime interest rate
2.2
Enter term number
3
Enter b for BusinessLoan & p for personal loan
p
Enter lastName
Vura
Enter loan number
12mbkg3425
Enter amount
40000
prime interest rate
1.2
Enter term number
1
Valiveti's loan number is 12bvgf2342 his loan amount is 90000.0 with intrest rate of 1.01 and total due is 90909.0 in term 1
Thota's loan number is 13vbjd5643 his loan amount is 70000.0 with intrest rate of 2.02 and total due is 71414.0 in term 1
Guntur's loan number is 14ncjd7865 his loan amount is 80000.0 with intrest rate of 2.02 and total due is 81616.0 in term 5
Sikakoli's loan number is 15vxgs5467 his loan amount is 50000.0 with intrest rate of 2.21 and total due is 51105.0 in term 3
Vura's loan number is 12mbkg3425 his loan amount is 40000.0 with intrest rate of 1.22 and total due is 40488.0 in term 1

D:\19BCD7088>
```