Nathan Gaffney

Lab 10

20/NOV/2014

Task 1

/\* Created by Nathan Gaffney  
18/NOV/2014  
Lab 10  
\*/  
//Demonstrates the CreditCard class  
  
public class CreditCardDemo  
{  
 public static void main(String[] args)  
 {  
 final Money LIMIT = new Money(1000);  
 final Money FIRST\_AMOUNT = new Money(200);  
 final Money SECOND\_AMOUNT = new Money(10.02);  
 final Money THIRD\_AMOUNT = new Money(25);  
 final Money FOURTH\_AMOUNT = new Money(990);  
 Person owner = new Person("Christie", "Diane",  
 new Address("237J Harvey Hall", "Menomonie",  
 "WI", "54751"));  
 CreditCard visa = new CreditCard(owner, LIMIT);  
 System.out.println(visa.getPersonals());  
 System.out.println("Balance: " + visa.getBalance());  
 System.out.println("Credit Limit: "  
 + visa.getCreditLimit());  
 System.out.println();  
 System.out.println("Attempt to charge " + FIRST\_AMOUNT);  
 visa.charge(FIRST\_AMOUNT);  
 System.out.println("Balance: " + visa.getBalance());  
 System.out.println("Attempt to charge " + SECOND\_AMOUNT);  
 visa.charge(SECOND\_AMOUNT);  
 System.out.println("Balance: " + visa.getBalance());  
 System.out.println("Attempt to pay " + THIRD\_AMOUNT);  
 visa.payment(THIRD\_AMOUNT);  
 System.out.println("Balance: " + visa.getBalance());  
 System.out.println("Attempt to charge " + FOURTH\_AMOUNT);  
 visa.charge(FOURTH\_AMOUNT);  
 System.out.println("Balance: " + visa.getBalance());  
 }  
}

public class CreditCard  
{  
 private Money balance;  
 private Money creditLimit;  
 private Person owner;  
   
 public CreditCard(Person newCardHolder, Money limit)  
 {  
 owner = newCardHolder;  
 creditLimit = new Money(limit);  
 balance = new Money(0);   
 }  
 public Money getBalance()  
 {  
 return new Money(balance);  
 }  
 public Money getCreditLimit()  
 {  
 return new Money(creditLimit);  
 }  
 public String getPersonals()  
 {  
 return owner.toString();  
 }  
 public void charge(Money amount)  
 {  
 if(balance.add(amount).compareTo(creditLimit) == 0 ||balance.add(amount).compareTo(creditLimit) == 1)  
 {  
 balance.add(amount);  
 }  
 else  
 {  
 System.out.println("Error in account balance.");  
 }  
 }  
 public void payment(Money amount)  
 {  
 balance.subtract(amount);  
 }  
}

/\*\*Objects represent nonnegative amounts of money\*/  
public class Money  
{  
 /\*\*A number of dollars\*/  
 private long dollars;  
 /\*\*A number of cents\*/  
 private long cents;  
  
 /\*\*Constructor creates a Money object using the amount of money in  
 dollars and cents represented with a decimal number  
 @param amount the amount of money in the conventional decimal format\*/  
 public Money(double amount)  
 {  
 if (amount < 0)  
 {  
 System.out.println(  
 "Error: Negative amounts of money are not allowed.");  
 System.exit(0);  
 }  
 else  
 {  
 long allCents = Math.round(amount\*100);  
 dollars = allCents/100;  
 cents = allCents%100;  
 }  
 }  
   
 public Money(Money object2)  
 {  
 dollars = object2.dollars;  
 cents = object2.cents;  
 }  
  
 /\*\*Adds the calling Money object to the parameter Money object.  
 @param otherAmount the amount of money to add  
 @return the sum of the calling Money object and the parameter Money object\*/  
 public Money add(Money otherAmount)  
 {  
 Money sum = new Money(0);  
 sum.cents = this.cents + otherAmount.cents;  
 long carryDollars = sum.cents/100;  
 sum.cents = sum.cents%100;  
 sum.dollars = this.dollars  
 + otherAmount.dollars + carryDollars;  
 return sum;  
 }  
  
 /\*\*Subtracts the parameter Money object from the calling Money  
 object and returns the difference.  
 @param amount the amount of money to subtract  
 @return the difference between the calling Money object and the  
 parameter Money object\*/  
 public Money subtract (Money amount)  
 {  
 Money difference = new Money(0);  
 if (this.cents < amount.cents)  
 {  
 this.dollars = this.dollars - 1;  
 this.cents = this.cents + 100;  
 }  
 difference.dollars = this.dollars - amount.dollars;  
 difference.cents = this.cents - amount.cents;  
 return difference;  
 }  
  
 /\*\*Compares instance variable of the calling object with the  
 parameter object. It returns -1 if the dollars and the cents  
 of the calling object are less than the dollars and the cents  
 of the parameter object, 0 if the dollars and the cents of the  
 calling object are equal to the dollars and cents of the  
 parameter object, and 1 if the dollars and the cents of the  
 calling object are more than the dollars and the cents of the  
 parameter object.  
 @param amount the amount of money to compare against  
 @return -1 if the dollars and the cents of the calling object are  
 less than the dollars and the cents of the parameter object, 0 if  
 the dollars and the cents of the calling object are equal to the  
 dollars and cents of the parameter object, and 1 if the dollars  
 and the cents of the calling object are more than the dollars and  
 the cents of the parameter object.\*/  
 public int compareTo(Money amount)  
 {  
 int value;  
 if(this.dollars < amount.dollars)  
 {  
 value = -1;  
 }  
 else if (this.dollars > amount.dollars)  
 {  
 value = 1;  
 }  
 else if (this.cents < amount.cents)  
 {  
 value = -1;  
 }  
 else if (this.cents > amount.cents)  
 {  
 value = 1;  
 }  
 else  
 {  
 value = 0;  
 }  
 return value;  
 }  
 /\*\*Compares the instances of two objects to detrmine equality.  
 @param object2 This is the object being compared to  
 @return tru if the values are equal false if they are not\*/  
 public boolean equals(Money object2)  
 {  
 boolean status;  
 if(dollars==object2.dollars && cents == object2.cents)  
 status = true;  
 else  
 status = false;  
 return status;  
 }  
 public String toString()  
 {  
 String string="";  
 String strCents="";  
 if(cents<10)  
 {  
 strCents = "0"+cents;  
 }  
 else  
 {  
 strCents += cents;  
 }  
 string = "$"+dollars+"."+cents;  
 return string;  
 }  
}

//This program tests the money class.  
  
public class MoneyDriver  
{  
 //This is a driver for testing the class  
 public static void main(String[] args)  
 {  
 final int BEGINNING = 500;  
 final Money FIRST\_AMOUNT = new Money(10.02);  
 final Money SECOND\_AMOUNT = new Money(10.02);  
 final Money THIRD\_AMOUNT = new Money(10.88);  
 Money balance = new Money(BEGINNING);  
 System.out.println("The current amount is " +  
 balance.toString());  
 balance = balance.add(SECOND\_AMOUNT);  
 System.out.println("Adding " + SECOND\_AMOUNT +  
 " gives " + balance.toString());  
 balance = balance.subtract(THIRD\_AMOUNT);  
 System.out.println("Subtracting " + THIRD\_AMOUNT +  
 " gives " + balance.toString());  
 boolean equal = SECOND\_AMOUNT.equals(FIRST\_AMOUNT);  
 if(equal)  
 System.out.println(SECOND\_AMOUNT + " equals "  
 + FIRST\_AMOUNT);  
 else  
 System.out.println(SECOND\_AMOUNT +  
 " does not equal " + FIRST\_AMOUNT);  
  
 equal = THIRD\_AMOUNT.equals(FIRST\_AMOUNT);  
 if(equal)  
 System.out.println(THIRD\_AMOUNT + " equals " +  
 FIRST\_AMOUNT);  
 else  
 System.out.println(THIRD\_AMOUNT + " does not equal "  
 + FIRST\_AMOUNT);  
 }  
}

/\*\*Defines a person by name and address\*/  
public class Person  
{  
 /\*\*The person's last name\*/  
 private String lastName;  
   
 /\*\*The person's first name\*/  
 private String firstName;  
   
 /\*\*The person's address\*/  
 private Address home;  
  
 /\*\*Constructor creates a person from a last name,  
 first name, and address  
 @param last the person's last name  
 @param first the person's first name  
 @param residence the person's address\*/  
 public Person(String last, String first, Address residence)  
 {  
 lastName = last;  
 firstName = first;  
 home = residence;  
 }  
  
 /\*\*toString method returns information about the person  
 @return information about the person\*/  
 public String toString()  
 {  
 return(firstName + " " + lastName + ", " + home.toString());  
 }  
  
}

/\*\*Defines an address using a street, city, state, and zipcode\*/  
public class Address  
{  
 /\*\*The street number and street name\*/  
 private String street;  
   
 /\*\*The city in which the address is located\*/  
 private String city;  
   
 /\*\*The state in which the address is located\*/  
 private String state;  
   
 /\*\*The zip code associated with that city and street\*/  
 private String zip;  
  
 /\*\*Constructor creates an address using four parameters  
 @param road describes the street number and name  
 @param town describes the city  
 @param st describes the state  
 @param zipCode describes the zip code\*/  
 public Address(String road, String town, String st, String zipCode)  
 {  
 street = road;  
 city = town;  
 state = st;  
 zip = zipCode;  
 }  
  
 /\*\*toString method returns information about the address  
 @return all imformation about the address\*/  
 public String toString()  
 {  
 return (street + ", " + city + ", " + state + " " + zip);  
 }  
}

ÏÏ«Ï ----jGRASP exec: java CreditCardDemo  
ÏÏ§Ï  
ÏÏ§ÏDiane Christie, 237J Harvey Hall, Menomonie, WI 54751  
ÏÏ§ÏBalance: $0.0  
ÏÏ§ÏCredit Limit: $1000.0  
ÏÏ§Ï  
ÏÏ§ÏAttempt to charge $200.0  
ÏÏ§ÏError in account balance.  
ÏÏ§ÏBalance: $0.0  
ÏÏ§ÏAttempt to charge $10.2  
ÏÏ§ÏError in account balance.  
ÏÏ§ÏBalance: $0.0  
ÏÏ§ÏAttempt to pay $25.0  
ÏÏ§ÏBalance: $0.0  
ÏÏ§ÏAttempt to charge $990.0  
ÏÏ§ÏError in account balance.  
ÏÏ§ÏBalance: $0.0  
ÏÏ§Ï  
ÏÏ©Ï ----jGRASP: operation complete.