一、　实验名称

More Classes and Objects

二、　实验目的

- Be able to write a copy constructor
- Be able to write equals and toString methods
- Be able to use objects made up of other objects (Aggregation)
- Be able to write methods that pass and return objects

三、　实验内容

Task #1 Writing a Copy Constructor
Task #2 Writing equals and toString methods
Task #3 Passing and Returning Objects

四、　实验方法(原理、流程图)

1. Written by Intellij IDEA Community edition 2020.3

2.

Task 1:

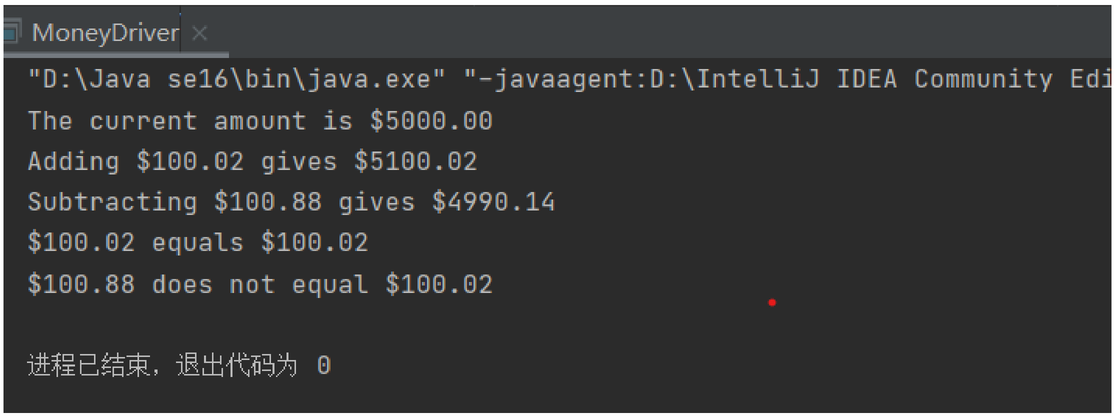
（1）Copy the files Address.java (code listing 8.1), Person.java (code listing 8.2),Money.java (code listing 8.3), MoneyDriver.java (code listing 8.4), and CreditCardDemo.java (code listing 8.5) from the Student CD or as directed by your instructor.
（2）Overload the constructor.Write a copy constructor. It should use the parameter money object to make a duplicate money object, by copying the value of each instance variable from the parameter object to the instance variable of the new object.
Task 2:
（1）Write and document an equals method. The method compares the instance variables of the calling object with instance variables of the parameter object for equality and returns true if the dollars and the cents of the calling object are the same as the dollars and the cents of the parameter object. Otherwise, it returns false.
（2）Write and document a toString method. This method will return a String that looks like money, including the dollar sign. Remember that if you have less than 10 cents, you will need to put a 0 before printing the cents so that it appears correctly with 2 decimal places.
Task 3:
(1) Create a CreditCard class according to the UML Diagram.
(2) Creat two parameters, a Person to initialize the owner and a Money value to initialize the creditLimit. The balance can be initialized to a Money value of zero.
(3) It should have accessor methods to get the balance and the available credit.
(4) It should have an accessor method to get the information about the owner, but in the form of a String that can be printed out.
(5) It should have a method that will charge to the credit card by adding the amount of Money in the parameter to the balance if it will not exceed the credit limit.

(6) It should have a method that will make a payment on the credit card by subtracting the amount of Money in the parameter from the balance.

五、　实验结论

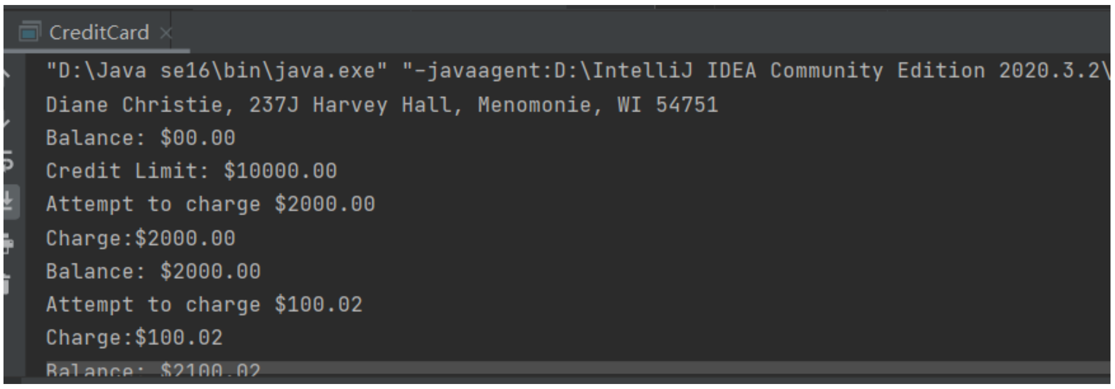
The experimental requirements have been successfully realized.

The given results are same as those calculated by my codes.

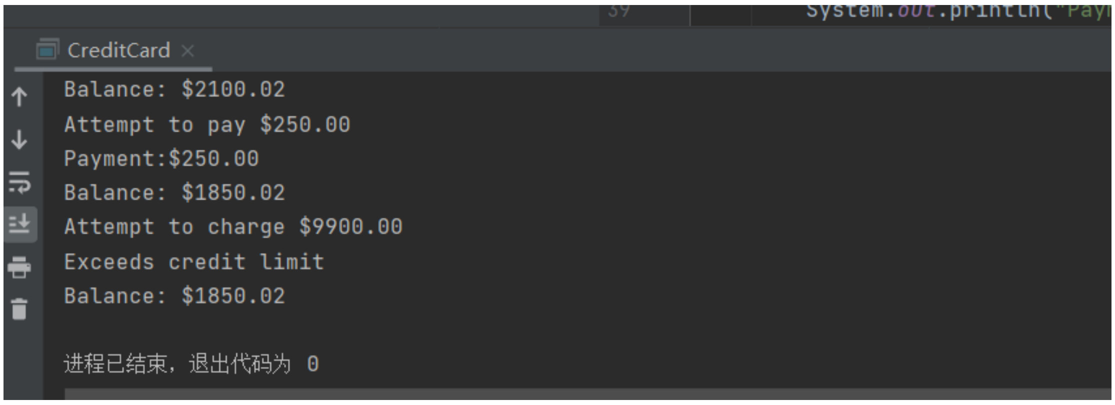


```
MoneyDriver
"D:\Java se16\bin\java.exe" "-javaagent:D:\IntelliJ IDEA Community Edi
The current amount is $5000.00
Adding $100.02 gives $5100.02
Subtracting $100.88 gives $4990.14
$100.02 equals $100.02
$100.88 does not equal $100.02

进程已结束，退出代码为 0
```



```
CreditCard
"D:\Java se16\bin\java.exe" "-javaagent:D:\IntelliJ IDEA Community Edition 2020.3.2\
Diane Christie, 237J Harvey Hall, Menomonie, WI 54751
Balance: $00.00
Credit Limit: $10000.00
Attempt to charge $2000.00
Charge:$2000.00
Balance: $2000.00
Attempt to charge $100.02
Charge:$100.02
Balance: $2100.02
```



```
CreditCard
Balance: $2100.02
Attempt to pay $250.00
Payment:$250.00
Balance: $1850.02
Attempt to charge $9900.00
Exceeds credit limit
Balance: $1850.02

进程已结束，退出代码为 0
```

六、实验体会和收获

1. By writing these code,I have a deepy realize about class and object.And this time i learn how to creat a copy constructor and it’sworking principle by myself.

2. When writing these codes,i meet some difficults.But I solve these problemss by search relevant information on the Internet and review PPT.

3. By writing this task I am more interested in Java.

七、程序代码

(1) Money. java

```
import java.text.DecimalFormat;

/**Objects represent nonnegative amounts of money*/

public class Money

{

    /**A number of dollars*/

    private long dollars;

    /**A number of cents*/

    private long cents;

    public Money(Money otherObject)

    {

        this.dollars = otherObject.dollars;

        this.cents = otherObject.cents;

    }

    @Override

    public boolean equals(Object object)

    {

        return this.dollars == ((Money)object).dollars && this.cents == ((Money)
object).cents;

    }

    @Override
    public String toString()
    {
        return "$"+dollars+String.format("%.2f",cents/100.0);
    }
    /**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;
        }
```

```java
    }

    /**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
        {
```

```java
                value = 0;
            }
            return value;
    }
}
```

(2) Creditcard. java

```java
/**Demonstrates the CreditCard class*/
public class CreditCard
{
    private Money balance = new Money(0);
    private Money creditLimit;
    private Person owner;
    public CreditCard(Person newCardHolder, Money limit)
    {
        this.owner = newCardHolder;
        this.creditLimit = limit;
    }
    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) != 1)
        {
            balance=balance.add(amount);
            System.out.println("Charge:"+amount);
        }
        else
        {
            System.out.println("Exceeds credit limit");
        }
    }
    public void payment(Money amount)
    {
        balance = balance.subtract(amount);
        System.out.println("Payment:"+amount);
    }
    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("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());

    }

}
```