

Java实验报告

Java实验报告

@Author : liyajun_208111545116

实验一 构建Java开发环境

一、实验目的

- 1、掌握Java 开发环境的建立和测试方法
- 2、掌握Java程序的编辑、编译以及运行方法
- 3、熟悉Java 基本编程

二、实验要求

1. JDK的安装设置
2. Eclipse/IDEA集成开发环境的安装
3. 掌握Java语言的基本知识：JDK、JRE、JVM等术语、Java源文件布局、程序入口main()函数、Java程序的编译运行
4. 简单的错误调试

三、实验内容

1. 下载并安装JDK;
2. 修改系统环境变量：path;
3. 使用文本文件编辑Java源程序，并使用JDK提供的Java工具包进行编译、运行;
4. 安装Eclipse/IDEA;
5. 使用集成开发环境（Eclipse/IDEA）编辑Java源程序文件并编译、运行。

搭建环境并输出Hello World

```
1 public static void main(String[] args){
2     // write your code here
3     System.out.println("Hello World");
4 }
```

```
.jar=55238:C:\Users\liyaj\AppData\Local\Jet
demo.demo01
Hello World

Process finished with exit code 0
```

实验二 Java基本编程

一、实验目的

- 1、熟悉Java 的标识符、关键字、数据类型、数组、枚举、命令行输入输出、分支与循环
- 2、掌握Java程序的编辑、编译以及运行方法
- 3、掌握Java 开发环境的建立和测试方法

二、预备知识

1. JDK的安装设置：JDK/JRE/JVM
2. Eclipse/IDEA集成开发环境的安装
3. 掌握Java语言的基本知识（Java类文件布局、程序入口main()函数、包、import语句、构造方法）
4. 掌握Java语言标识符、关键字、数据类型、数组、枚举、命令行输入输出、分支与循环
5. 简单的程序错误调试

三、实验描述

1. 实验类型：设计
2. 实验学时：2学时
3. 实验内容：1项（参照第四项）

四、实验步骤

(1) 检查所用的计算机系统：

1. 确认是否已安装JDK，并确认系统的环境变量设置；
2. 确认是否已安装Eclipse/IDEA集成开发环境。

(2) 实验内容：设计Java源程序，并作简单地运行测试

具体要求：

1. 编写一个Java应用程序。用户从键盘输入一个1 ~ 99999之间的数，程序将判断这个数是几位数，并判断这个数是否是回文数。回文数是指将该数含有的数字逆序排列后得到的数和原数相同，例如12121、3223都是回文数。
2. 编写一个Java应用程序，实现下列功能：

·程序随机分配给客户一个1 ~ 100之间的整数。（提示：使用Random类）

```
1 Random random = new Random();
2 int realNumber = random.nextInt(100) + 1;
```

·用户反复输入自己的猜测并进行比较。

·程序返回提示信息，分别是“猜大了”、“猜小了”或“猜对了”，直到猜对为止。

3. 一个数如果恰好等于它的因子之和，这个数就称为“完数”。编写一个应用程序，求1000以内的所有完数。

回文数 ::

思路 ::

头尾双向扫描 时间复杂度 $O(n)$

代码实现 ::

```
1 boolean valid(int i) {
2     String s = Integer.toString(i);
3     int l = 0, r = s.length() - 1;
4     while(l < r) {
5         if(s.toCharArray()[l++] != s.toCharArray()[r--]) return false;
6     }
7     return true;
8 }
```

结果 ::

```
1 12121 -> true;
2 114514 -> false;
```

猜数字 ::

思路 ::

二分出来一个答案 时间复杂度 $O(\log n)$

代码实现 ::

```

1 public void guess(){
2     Random ra = new Random();
3     int cur = ra.nextInt(100) + 1;
4     System.out.println("please enter a num");
5     Scanner sc = new Scanner(System.in);
6     int get = sc.nextInt();
7     while(get != cur){
8         if(get > cur) System.out.println("Try a smaller number");
9         else if(get < cur) System.out.println("Try a larger number");
10        get = sc.nextInt();
11    }
12    System.out.println("congratulations");
13 }

```

结果 ::

```

C:\Users\liya\AppData\Local\JetBrains\Toolbox
demo.demo01
please enter a num
50
Try a larger number
85
Try a smaller number
65
Try a larger number
75
congratulations

Process finished with exit code 0

```

完数 ::

思路 ::

根据定义暴力求解

代码实现 ::


```

1 public static void main(String[] args){
2     // write your code here
3     for(int i = 1; i <= 1000; i++){
4         if(i % 2 == 0 && valid(i)) {
5             System.out.println(i);
6         }
7     }
8 }
9 boolean valid(int x) {

```

```
10     int cur = x;
11     for(int i = 1; i <= x/2; i++) {
12         if(x % i == 0) cur -= i;
13     }
14     return cur == 0;
15 }
```

运行结果 ::



```
6
28
496
```

总结 ::

使用java的一些语言特性来进行程序设计

实验三 银行账户管理 - 面向对象 ::

一、实验目的 ::

- 1、熟悉Java 开发环境的建立和测试方法
- 2、掌握Java类的构造，使用类来封装对象的属性和功能
- 3、熟悉Java程序的编辑、编译以及运行方法

二、预备知识 ::

1. JDK的安装设置
2. Eclipse/IDEA集成开发环境的安装;
3. Java类与对象的基本知识 (Java类文件布局、方法重载、构造方法、static关键字、this关键字、包与import语句、访问权限等)
4. 简单的程序错误调试

三、实验描述 ::

1. 实验类型：设计
2. 实验学时：2学时
3. 实验内容：1项（参照第四项）

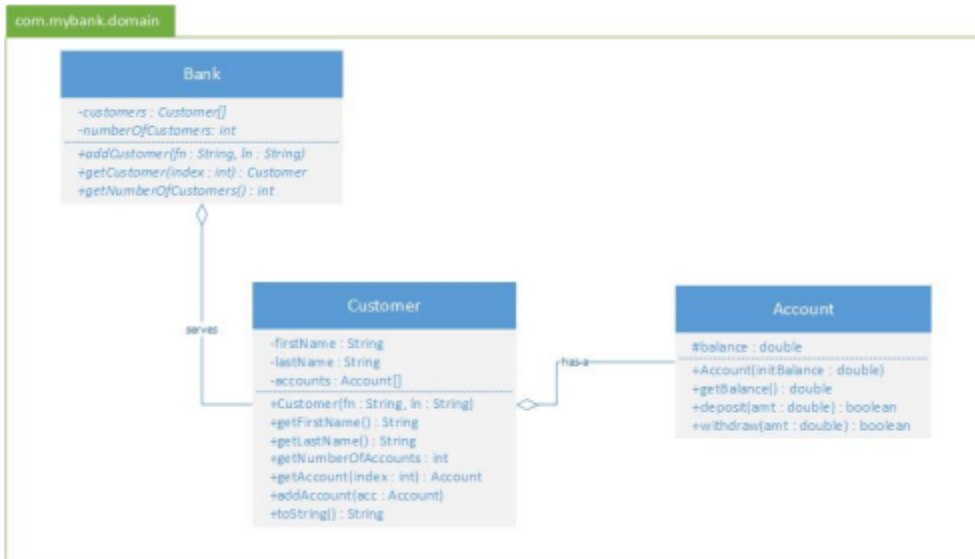
四、实验步骤 ::

- (1) 检查所使用的计算机系统：

1. 确认是否已安装JDK，并确认系统的环境变量设置；
2. 确认是否已安装Eclipse/IDEA集成开发环境。

(2) 实验内容：设计Java类文件，并作简单测试

具体要求：



1. 创建项目：BankProject;
2. 创建Account类:

·位于包：com.mybank.domain;

·向Account类中添加protected实例变量balance，用于维护客户的账户存款额；

·添加具有一个参数的公有构造方法：initBalance，用于初始化账户存款额；

·添加一个公有方法：deposit(double amt)，用于向账户存钱，返回值是boolean型；

·添加一个公有方法：withdraw(double amt)，用于从账户取钱，需要根据账户的实际余额做判断分析，返回值是boolean型；

·添加一个公有方法：getBalance()，用于返回balance的值，返回值是double类型。

3. 创建Customer类:

·位于包：com.mybank.domain中;

·向Customer类添加四个实例变量：firstName，lastName，accounts（Account对象数组，长度为10）和numberOfAccounts（记录实际的账户数目）；

·添加一个公有构造方法：具有两个形式参数firstName，lastname，用于初始化客户姓名；同时初始化accounts数组和numberOfAccounts变量；

- 添加getFirstName方法：该方法返回客户的firstName实例变量；
- 添加getLastName方法：该方法返回客户的lastName实例变量；
- 添加addAccount方法：该方法具有一个形式参数：double amount，数组的下标通过实例变量numberOfAccounts实现递增，同时将实例化出来的Account对象存储到accounts数组中；
- 添加重载的addAccount(Account acc)方法：该方法具有一个形式参数（即Account类型的对象acc），数组的下标依然通过实例变量numberOfAccounts实现递增，同时将Account对象acc存储到accounts数组中；
- 添加getNumberOfAccounts方法：该方法返回numberOfAccounts实例变量；
- 添加getAccount方法：该方法具有一个形式参数（即int类型index），根据传进来的数组下标返回该下标对应的Account对象。
- 重写toString() 方法，将Customer对象以客户名字的字符串形式输出。

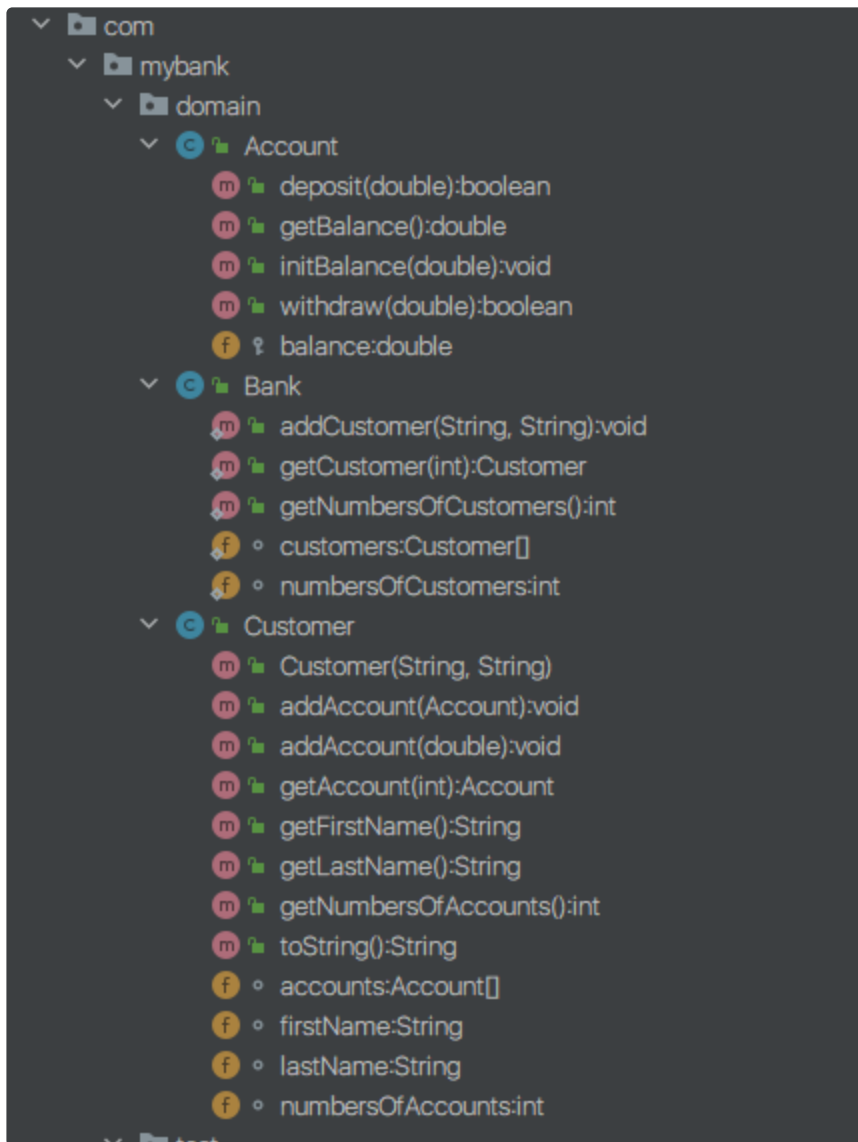
4. 创建Bank类：

- 位于包：com.mybank.domain中；
- 向Bank类添加两个静态变量：customers（Customer对象数组，长度为10）和numberOfCustomers（记录实际客户数目）；
- 添加一个静态初始化块：用于初始化customers数组和numberOfCustomers变量；
- 添加静态方法addCustomer：该方法有两个形式参数（即firstName，lastName），数组的下标通过实例变量numberOfCustomers实现递增，同时将实例化出来的Customer对象存储到customers数组中；
- 添加静态方法getNumberOfCustomers：该方法返回numberOfCustomers的值；
- 添加静态方法getCustomer：该方法具有一个形式参数（即int类型index），根据传进来的数组下标返回该下标对应的Customers对象。

5. 创建TestBanking类，该类有程序入口 main() 函数：

- 该类文件位于包：com.mybank.test；
- 通过Bank类调用相应的静态方法，为银行的每个客户及其账户开户；
- 使用双重for循环遍历银行类的customers数组以及每个客户的accounts数组，通过随机数类（Random类）或者Math类的random()方法，随机地为每个账户分别完成存取款操作，要求保留两位小数，并最终打印输出客户姓名及其每个账户的余额。

项目结构 ::



实现代码 ::

Account.java ::

```
1 package com.mybank.domain;
2
3 public class Account {
4     protected double balance;
5     //初始化余额
6     public Account(double balance){
7         this.balance = balance;
8     }
9     //存钱
10    public boolean deposit(double amt){
11        if(amt < 0) return false;
12        balance += amt;
13        return true;
14    }
15    //取钱
16    public boolean withdraw(double amt){
```



```

17         if(balance < amt) return false;
18         balance -= amt;
19         return true;
20     }
21
22     //获取余额
23     public double getBalance(){
24         return balance;
25     }
26 }
27

```

Bank.java

```

1 package com.mybank.domain;
2
3 public class Bank {
4     static Customer [] customers;
5     static int numbersOfCustomers;
6     static {
7         customers = new Customer[10];
8         numbersOfCustomers = 0;
9     }
10    public static void addCustomer(String firstName, String lastName){
11        Customer cus = new Customer(firstName,lastName);
12        try {
13            customers[numbersOfCustomers++] = cus;
14        } catch (ArrayIndexOutOfBoundsException e){
15            System.out.println(e);
16        }
17    }
18    public static int getNumbersOfCustomers(){
19        return numbersOfCustomers;
20    }
21    public static Customer getCustomer(int id){
22        try {
23            return customers[id];
24        } catch (ArrayIndexOutOfBoundsException e){
25            System.out.println(e);
26        }
27        return null;
28    }
29 }
30

```

Customer.java

```
1 package com.mybank.domain;
2
3 public class Customer {
4     String firstName;
5     String lastName;
6     Account [] accounts;
7     int numbersOfAccounts;
8
9     public Customer(String firstName, String lastName) {
10         this.firstName = firstName;
11         this.lastName = lastName;
12         numbersOfAccounts = 0;
13         accounts = new Account[10];
14     }
15     public void addAccount(double amount){
16         addAccount(new Account(amount));
17     }
18     public void addAccount(Account acc){
19         try {
20             accounts[numbersOfAccounts++] = acc;
21         }catch (ArrayIndexOutOfBoundsException e){
22             System.out.println(e);
23         }
24     }
25     public Account getAccount(int id){
26         try {
27             return accounts[id];
28         }catch (ArrayIndexOutOfBoundsException e){
29             System.out.println(e);
30         }
31         return null;
32     }
33     public int getNumbersOfAccounts() {
34         return numbersOfAccounts;
35     }
36     public String getFirstName() {
37         return firstName;
38     }
39
40     public String getLastName() {
41         return lastName;
42     }
43
44     @Override
45     public String toString() {
46         return firstName + "-" + lastName ;
47     }
```

```
48 }  
49
```

TestBanking.java ::

```
1 package com.mybank.test;  
2  
3 import com.mybank.domain.Bank;  
4  
5 import java.util.Random;  
6  
7 public class TestBanking {  
8     public static void main(String[] args) {  
9         //初始化成员  
10        for(int i = 0; i < 10; i++){  
11            Random ra = new Random();  
12            Bank.addCustomer(Integer.toString(i), "Jinx");  
13            for(int j = 0; j < 10; j++){  
14                Bank.getCustomer(i).addAccount(ra.nextDouble(1145.14));  
15                Bank.getCustomer(i).getAccount(j).withdraw(ra.nextDouble(1145.14));  
16            }  
17        }  
18        for(int i = 0; i < 10; i++){  
19            System.out.println(Bank.getCustomer(i).toString());  
20            for(int j = 0; j < 10; j++){  
21                System.out.println("第" + (j + 1) + "个账户的余额为");  
22                System.out.println(String.format("%.2f", Bank.getCustomer(i).getAccount(j).getBalance()));  
23            }  
24        }  
25    }  
26 }  
27 }  
28
```

测试结果 ::

```
C:\Users\liyaj\.jdk\openjdk-17.0.2\bin\java.exe -javaagent:C:\Users\liyaj\AppData\Local\JetBrains\Toolbox\apps\IDEA-U\ch
.jar=57769:C:\Users\liyaj\AppData\Local\JetBrains\Toolbox\apps\IDEA-U\ch-0\213.7172.25\bin -Dfile.encoding=UTF-8 -classp
com.mybank.test.TestBanking
0-Jinx
第1个账户的余额为
72.67
第2个账户的余额为
184.98
第3个账户的余额为
30.48
第4个账户的余额为
57.66
第5个账户的余额为
721.69
第6个账户的余额为
191.60
第7个账户的余额为
764.83
第8个账户的余额为
454.07
第9个账户的余额为
223.04
第10个账户的余额为
22.55
1-Jinx
第1个账户的余额为
320.37
第2个账户的余额为
625.96
第3个账户的余额为
23.68
第4个账户的余额为
187.50
第5个账户的余额为
123.28
第6个账户的余额为
700.84
第7个账户的余额为
42.48
```

总结 ::

面向对象程序设计 熟悉和理解了Java类的构造，使用类来封装对象的属性和功能，和*Java*的运行调试

实验四 银行账户管理 – 继承与转型对象 .

一、实验目的 ::

1. 掌握Java 面向对象编程技术
2. 能够使用继承创建子类并实现方法覆盖
3. 掌握和理解向上转型和向下转型的概念和方法
4. 能够创建异类集合并使用多态

二、预备知识 ::

1. JDK的安装设置
2. Eclipse/IDEA集成开发环境的安装
3. 掌握Java语言的面向对象特性（封装性、继承性、多态性）

4. 类层次结构中向上转型和向下转型

三、实验描述

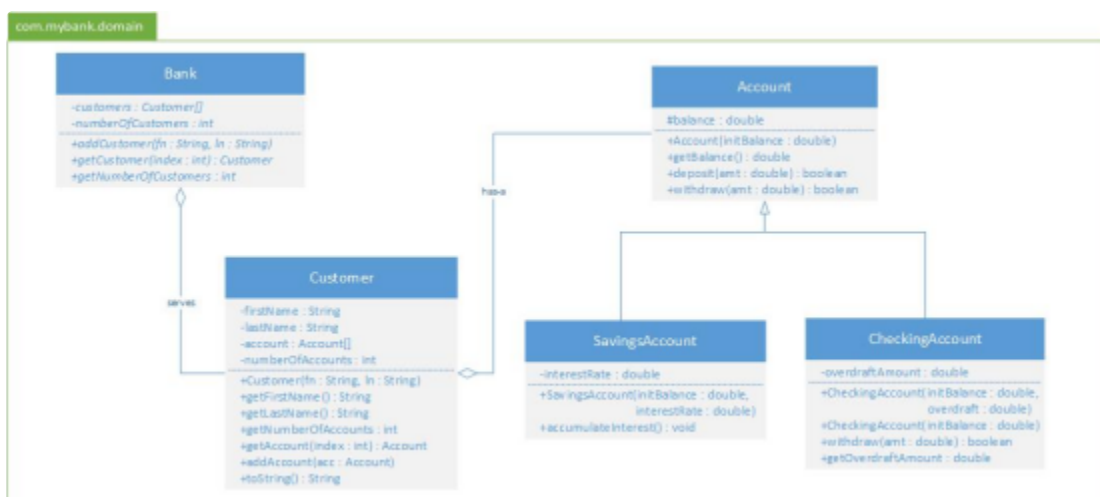
1. 实验类型：设计
2. 实验学时：2学时
3. 实验内容：1项（参照第四项）

四、实验内容

(1) 检查所使用的计算机系统：

1. 确认是否已安装JDK，并确认系统的环境变量设置；
2. 确认是否已安装Eclipse/IDEA集成开发环境。

(2) 实验内容：要求使用继承创建子类，并实现方法覆盖/重写，并能够创建异类集合测试类的多态特性。具体要求：



1. 创建SavingsAccount类，该类是Account类的子类：

·位于包：com.mybank.domain中；

·向SavingsAccount类中添加私有的实例变量：double interestRate；

·添加一个含有两个参数的公有构造方法，参数是：initBalance和interestRate。调用父类构造方法传递initBalance参数，并初始化实例变量interestRate；

·添加accumulateInterest方法：用于计算客户的利息。

2. 创建CheckingAccount类，该类是Account类的子类：

·位于包：com.mybank.domain中；

- 向CheckingAccount类中添加overdraftAmount实例变量;
- 添加一个有两个参数的公有构造方法, 参数是initBalance和overdraftAmount。调用父类构造方法传递initBalance参数, 并初始化实例变量overdraftAmount;
- 添加另一个具有一个参数initBalance的公有构造方法。使用initBalance参数和overdraftAmount参数调用第一个构造方法, 其中overdraftAmount参数使用默认值0.0;
- 添加一个公有方法: getOverdraftAmount(), 用于返回overdraftAmount的值, 返回值是double类型;
- 覆盖/重写withdraw方法, 方法参数为amount, balance变量继承于父类, 可见性protected, 所以子类可以直接使用。下面是withdraw方法的实现伪码:

```
1      if (balance < amount) then
2
3          double overdraftNeeded = amount - balance  // 计算还差多少钱
4
5          // 如果允许透支的额度小于差的钱, 交易终止
6
7          if (overdraftAmount < overdraftNeeded)
8
9              then transaction fails
10
11         else
12
13             // 否则先清空卡内余额, 并计算下一次允许透支的额度
14
15             balance = 0.0
16
17             decrement overdraftAmount by overdraftNeeded
18
19     else
20
21         // 卡内余额足够支付
22
23         decrement balance by amount
24
```

3. 创建TestBanking2类:

- 该类文件位于包: com.mybank.test
- 该类有程序入口 main() 函数;
- 借助已定义的Bank类的静态方法完成银行数据的初始化;

·使用双重for循环遍历银行类的customers数组以及每个客户的accounts数组，然后依次访问每一个账户Account对象，并根据不同的账户类型，通过向下转型操作，恢复具体的账户类型，从而做出不同的存、取钱操作。（例如：如果是SavingsAccount对象，则不能透支，但可以计算利息；如果是CheckingAccount对象，则可以进行适当额度的透支；如果是Account对象，那么就是普通的账户类型）；

·通过随机数类（Random类）完成每个账户的存取款功能，并最终打印输出客户姓名、账户类型、账户余额和允许透支的额度。

代码实现 ::

CheakAccount.java ::

```
1 package experiment.four.com.mybank.domain;
2 //能够透支的账户
3 public class CheckingAccount extends Account{
4     double overdraftAmount;
5
6     public CheckingAccount(double initBalance, double overdraftAmount){
7         super(initBalance);
8         this.overdraftAmount = overdraftAmount;
9     }
10    CheckingAccount(double initBalance){
11        super(initBalance);
12        this.overdraftAmount = 0.0;
13    }
14
15    public double getOverdraftAmount() {
16        return overdraftAmount;
17    }
18
19    @Override
20    public int withdraw(double amount) {
21        if(balance < amount){
22            //使用贷款
23            double overdraftNeeded = amount - balance;
24            if(overdraftNeeded > overdraftAmount){
25                return 1;
26            }
27            else{
28                balance = 0.0;
29                overdraftAmount -= overdraftNeeded;
30                return 2;
31            }
32        } // 能够付清
```

```

33         else{
34             balance -= amount;
35         }
36         return 3;
37     }
38 }
39

```

SavingAccount.java

```

1  package experiment.four.com.mybank.domain;
2  //能计算利息的账户
3  public class SavingsAccount extends Account{
4      private double interestRate;
5      public SavingsAccount(double initBalance,double interestRate){
6          super(initBalance);
7          this.interestRate = interestRate;
8      }
9
10
11     //计算利息
12     public double accumulateInterest(){
13         return balance + balance * interestRate;
14     }
15 }
16

```

Test

```

1  package experiment.four.com.mybank.test;
2
3
4  import experiment.four.com.mybank.domain.*;
5
6
7  public class TestBanking2 {
8      public static void main(String[] args) {
9          Bank.addCustomer("zhang", "san");
10         Customer c1 = Bank.getCustomer(0);
11         c1.addAccount(new SavingsAccount(500,0.017));
12         c1.addAccount(new CheckingAccount(2000, 3000));
13         for(int i = 0;i < Bank.getNumbersOfCustomers();i++) {
14             Customer cus = Bank.getCustomer(i);
15             System.out.println(cus);
16             for(int j = 0;j < cus.getNumbersOfAccounts();j++) {
17                 Account acc = cus.getAccount(j);

```

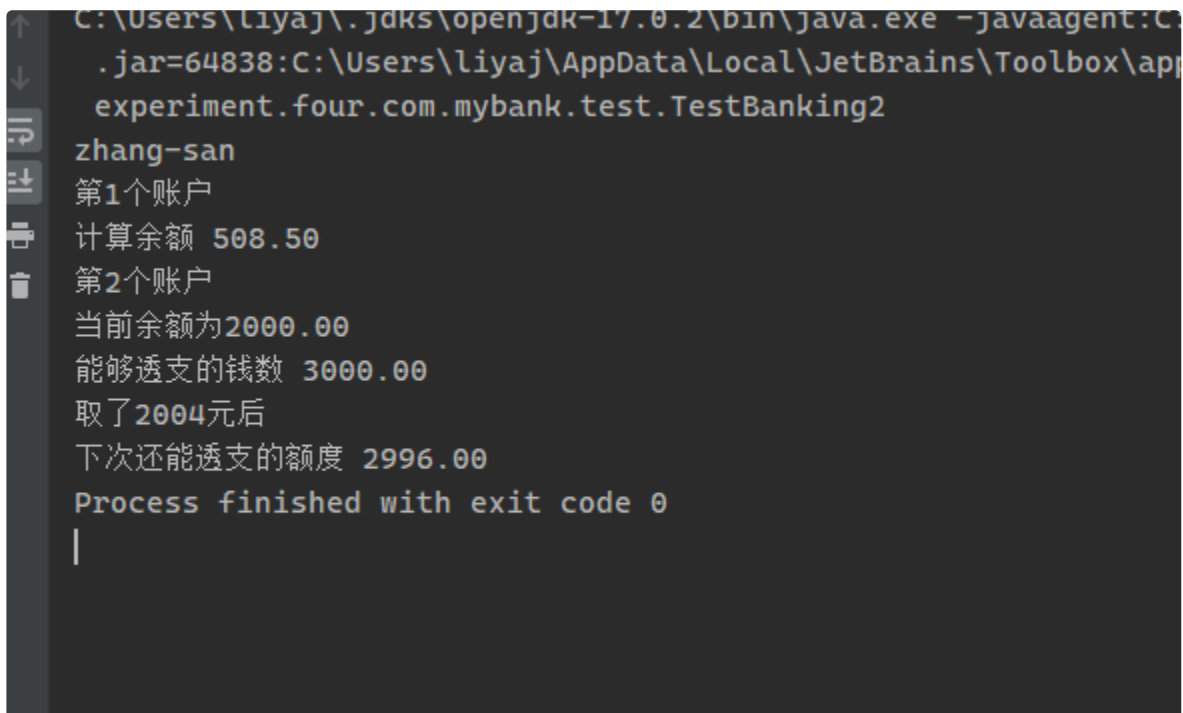


```

18         System.out.println("第" + (j + 1) + "个账户");
19         if (acc instanceof SavingsAccount sa) {
20             System.out.printf("计算余额 %.2f\n", sa.accumulateInterest());
21         }
22         else if (acc instanceof CheckingAccount ck) {
23             System.out.printf("当前余额为%.2f\n", ck.getBalance());
24             System.out.printf("能够透支的钱数 %.2f\n", ck.getOverdraftLimit());
25             ck.withdraw(2004);
26             System.out.println("取了2004元后");
27             System.out.printf("下次还能透支的额度 %.2f", ck.getOverdraftLimit());
28         }
29         else {
30             // 普通账户
31             System.out.printf("余额为 %.2f", acc.getBalance());
32         }
33     }
34 }
35 }
36 }
37

```

测试 ::



```

C:\Users\liyaj\.jdk\openjdk-17.0.2\bin\java.exe -javaagent:C:\Users\liyaj\AppData\Local\JetBrains\Toolbox\apps\IDEA-U\bin\idea-231.0.170-idea-231.jar=64838:C:\Users\liyaj\AppData\Local\JetBrains\Toolbox\apps\IDEA-U\bin\idea-231.0.170-idea-231.jar experiment.four.com.mybank.test.TestBanking2
zhang-san
第1个账户
计算余额 508.50
第2个账户
当前余额为2000.00
能够透支的钱数 3000.00
取了2004元后
下次还能透支的额度 2996.00
Process finished with exit code 0
|

```

总结 ::

掌握和理解了 *Java* 向上转型和向下转型的概念和方法，也理解了如何创建异类集合并使用多态

实验五 面向对象高级编程 — 动物宠物程序 ::

一、实验目的 ::

1. 掌握Java 面向对象编程技术
2. 能够创建抽象类和接口，并探究它们的多态特性

二、预备知识 ::

1. JDK的安装设置
2. Eclipse/IDEA集成开发环境的安装
3. 掌握Java语言的面向对象特性（封装性、继承性、多态性）
4. 掌握Java语言的抽象类和接口

三、实验描述 ::

1. 实验类型：设计
2. 实验学时：2学时
3. 实验内容：1项（参照第四项）

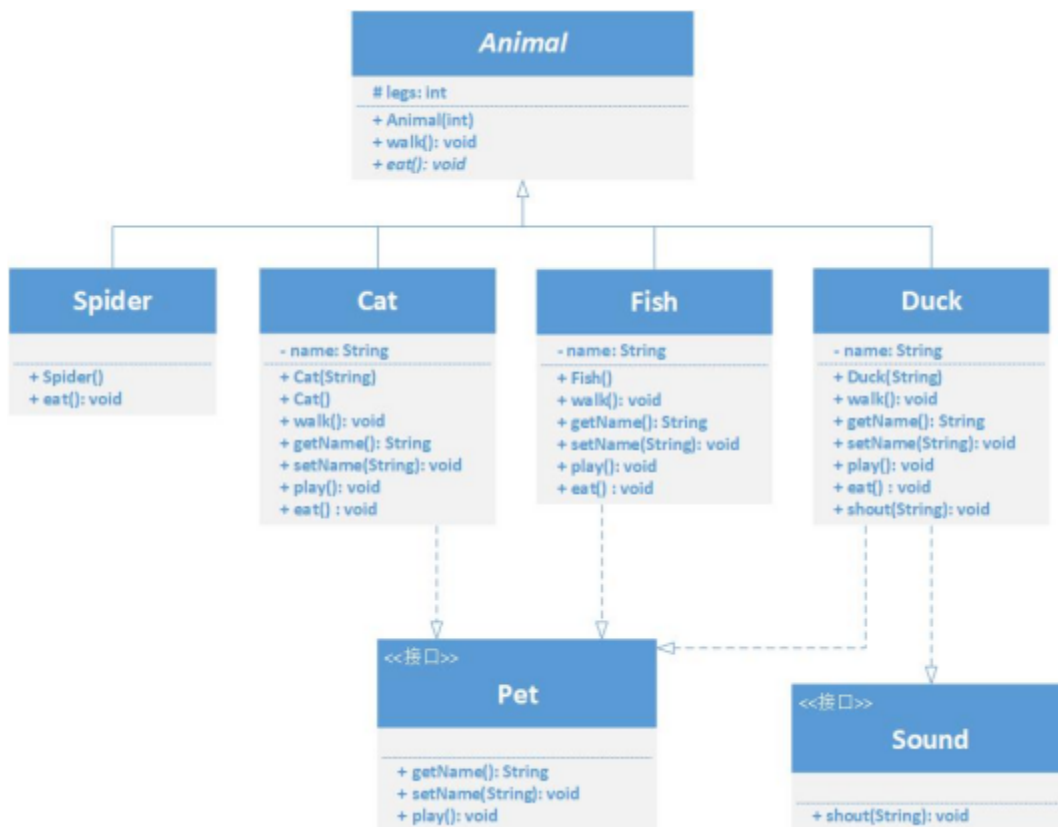
四、实验内容 ::

(1) 检查所使用的计算机系统：

1. 确认是否已安装JDK，并确认系统的环境变量设置；
2. 确认是否已安装Eclipse/IDEA集成开发环境。

(2) 实验内容：要求使用抽象类和接口创建不同的对象，并探究它们的多态特性

具体要求：



1. 创建项目：InterfaceProject;
2. 创建Animal类，该类是抽象类：

·声明一个受保护的整型实例变量legs，记录动物的腿的数目；

·定义一个受保护的构造方法初始化legs实例变量；

·声明抽象方法eat();

·声明具体方法walk，用于显示与动物行走方式有关的信息（包括腿的数目）。

3. 创建Pet接口，具有三个抽象方法：

```
public String getName();
```

```
public void setName(String n);
```

```
public void play();
```

4. 创建Sound接口，声明一个默认方法 shout()，该方法有具体实现，表示发出的声音：

```
public default void shout(String voice);
```

5. 创建Spider类：

·Spider类扩展了Animal类；

·定义一个无参数构造方法，调用父类构造方法来指明所有蜘蛛都有八条腿；

- 覆盖/重写抽象方法eat()。

6. 创建Cat类，扩展了Animal类，同时实现Pet接口：

- 声明String实例变量name存储宠物的名字；

- 定义一个构造方法，使用String类型的参数指定猫的名字；该构造方法必须调用父类构造方法来指明所有猫都有四条腿；

- 另外定义一个无参数的构造方法，它调用上面的一个构造方法（使用this关键字）来传递一个空字符串作为参数；

- 实现Pet接口的所有抽象方法；

- 覆盖/重写抽象方法eat()。

7. 创建Fish类，扩展了Animal类，同时实现Pet接口：

- 声明String实例变量name存储宠物的名字；

- 定义一个具有String类型参数的构造方法，它调用父类构造方法来指明鱼没有腿；

- 实现Pet接口的所有抽象方法；

- 覆盖/重写walk()方法，使用super关键字调用父类的walk() 方法，并打印输出一条说明鱼不会行走的消息；

- 覆盖/重写抽象方法eat()。

8. 创建Duck类，扩展了Animal类，同时实现Pet接口和Sound接口：

- 声明String实例变量name存储宠物的名字；

- 定义一个具有String类型参数的构造方法，它调用父类构造方法来指明鸭子有2条腿；

- 实现Pet接口的所有抽象方法；

- 覆盖/重写抽象方法eat()；

- 覆盖/重写Sound接口的默认方法shout()，并要求访问Sound接口的默认方法shout()。

9. 创建TestAnimal类：

- 该类具有程序入口main()函数；

- 创建并操作前面所创建的类的实例；

- 调用每个对象中的各个方法；

- 测试对象类型的转换；

·观察对象的多态特性。

动物宠物程序 ::

实现代码 ::

Animal.java ::

```
1 package experiment.five.InterfaceProject;
2
3 public abstract class Animal {
4     protected int legs;
5     protected Animal(int legs){
6         this.legs = legs;
7     }
8     abstract void eat();
9     public void walk(){
10         System.out.println("Animal walk with"+legs+"legs.");
11     }
12 }
13
```

Cat.java ::

```
1 package experiment.five.InterfaceProject;
2
3 public class Cat extends Animal implements Pet{
4     private String name;
5     public Cat(String name){
6         super(4);
7         this.name = name;
8     }
9     public Cat(){
10         this("");
11     }
12     @Override
13     void eat() {
14         System.out.println("Cat likes eating mice.");
15     }
16
17     @Override
18     public String getName() {
19         return this.name;
20     }
21
22     @Override
23     public void setName(String n) {
```

```

24         this.name = name;
25     }
26
27     @Override
28     public void play() {
29         System.out.println("Cat likes playing strings.");
30     }
31 }

```

Duck.java

```

1  package experiment.five.InterfaceProject;
2
3  public class Duck extends Animal implements Pet, Sound {
4      private String name;
5
6      public Duck( String name) {
7          super(2);
8          this.name = name;
9      }
10
11     @Override
12     public void walk() {
13         System.out.println("duck swim and walks with"+legs+"legs");
14     }
15
16     @Override
17     void eat() {
18         System.out.println("duck like eating waterweeds");
19     }
20
21     @Override
22     public String getName() {
23         return this.name;
24     }
25
26     @Override
27     public void setName(String n) {
28         this.name = name;
29     }
30
31     @Override
32     public void play() {
33         System.out.println("duck likes playing with water");
34     }
35     public void shout(String voice){
36         Sound.super.shout(voice);

```

```

37         System.out.println("i am a yellow duck");
38     }
39 }

```

Fish.java

```

1  package experiment.five.InterfaceProject;
2
3  public class Fish extends Animal implements Pet{
4      private String name;
5      @Override
6      void eat() {
7          System.out.println("Fish likes eating bugs in the ponds");
8      }
9
10     public Fish() {
11         super(0);
12     }
13
14     @Override
15     public String getName() {
16         return this.name;
17     }
18
19     @Override
20     public void setName(String n) {
21         this.name = n;
22     }
23
24     @Override
25     public void play() {
26         System.out.println("fish likes swimming in yje ponds");
27     }
28     @Override
29     public void walk(){
30         super.walk();
31         System.out.println("fish are swimming happily in the pool.");
32     }
33 }

```

Pet.interface

```

1  package experiment.five.InterfaceProject;
2
3  public interface Pet {
4      public String getName();

```

```
5     public void setName(String n);
6     public void play();
7 }
```

Sound.interface ::

```
1 package experiment.five.InterfaceProject;
2
3 public interface Sound {
4     public default void shout(String voice){
5         System.out.println(voice);
6     }
7 }
```

Spider.java ::

```
1 package experiment.five.InterfaceProject;
2
3 public class Spider extends Animal{
4     public Spider(){
5         super(8);
6     }
7
8     @Override
9     void eat() {
10         System.out.println("Spider eats flying on the net");
11     }
12 }
```

Test ::

```
1 package experiment.five.InterfaceProject;
2
3 public class TestAnimal {
4     public static void main(String[] args) {
5         Spider s1 = new Spider();
6         s1.eat();
7         Animal s2 = new Cat();
8         System.out.println(s2.legs);
9         s2.walk();
10        s2.eat();
11        Pet s3 = new Cat("fluffy");
12        s3.getName();
13        s3.play();
14        Cat s4 = new Cat("Grafield");
15        s4.getName();
16    }
17 }
```



```

16         int legs = s4.legs;
17         s4.eat();
18         s4.walk();
19         s4.play();
20         Pet s5 = new Fish();
21         s5.getName();
22         Duck s6 = new Duck("donald duck");
23         s6.shout("gagaga");
24     }
25 }

```

测试 ::

```

C:\Users\liyaj\.jbrs\openjdk-17.0.2\bin\java.exe -java
.jar=53542:C:\Users\liyaj\AppData\Local\JetBrains\Tool
experiment.five.InterfaceProject.TestAnimal
Spider eats flying on the net
4
Animal walk with4legs.
Cat likes eating mice.
Cat likes playing strings.
Cat likes eating mice.
Animal walk with4legs.
Cat likes playing strings.
gagaga
i am a yellow duck

```

::

学习了Java的接口，学会了创建抽象类和接口，并探究它们的多态特性

实验六 银行账户管理 – 异常处理及泛型容器 ::

一、实验目的 ::

1. 熟练掌握Java 面向对象的编程
2. 能够定义、创建并使用异常类
3. 能够定义、创建并使用异常类

二、预备知识 ::

1. JDK的安装设置
2. Eclipse/IDEA集成开发环境的安装
3. 掌握Java语言的面向对象特性（封装性、继承性、多态性）

- 3.
4. 掌握Java应用编程关于泛型容器、异常处理机制等相关知识

三、实验描述

1. 实验类型：设计
2. 实验学时：2学时
3. 实验内容：1项（参照第四项）

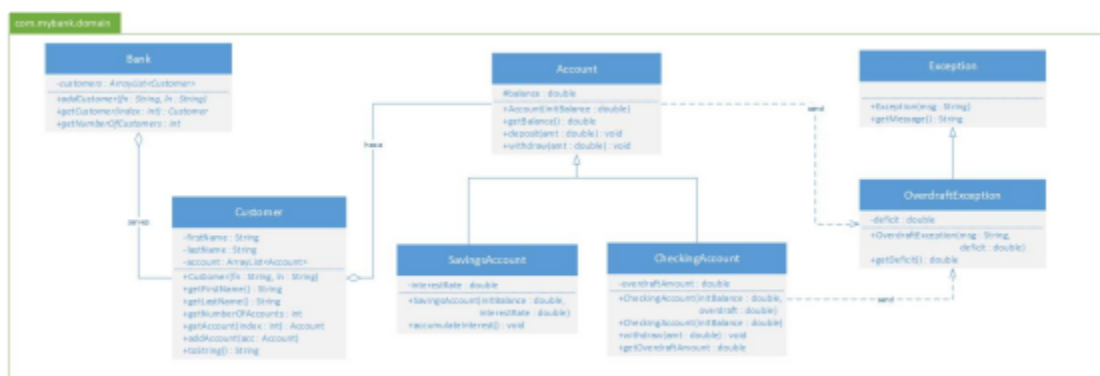
四、实验内容

(1) 检查所使用的计算机系统：

1. 确认是否已安装JDK，并确认系统的环境变量设置；
2. 确认是否已安装Eclipse/IDEA集成开发环境。

(2) 实验内容：设计Java类文件，并作简单测试

具体要求：



1. 继续修改项目：BankProject

- 创建OverdraftException异常类，扩展Exception类；
- 声明double类型的私有实例变量：deficit，代表赤字；
- 添加一个公有构造方法，具有两个参数：message和deficit。message参数通过super语句传递给父类的构造方法，deficit参数用于初始化deficit实例变量；
- 声明一个公有方法getDeficit()，返回deficit实例变量的值。

2. 修改Account类：

- 修改deposit()方法的返回值为void；
- 修改withdraw()方法的返回值为void，当余额不足时，该方法会抛出OverdraftException异常，抛出异常时需要指明原因和赤字的额度。

3. 同样的方式，修改CheckingAccount类：

- 修改withdraw()方法的返回值为void；

- 当OverdraftAmount数额不足以弥补赤字时，修改代码以抛出异常，需要指明原因和赤字的额度。

4. 顺着编译报错去修改上级调用者TestBanking2类：此处选择通过try-catch捕获异常，在异常处理中指出异常发生的原因和赤字额度。

5. 修改Customer类：

- 位于包com.mybank.domain中；

- 将accounts实例变量的声明修改为List<Account>类型，使用泛型，不再使用numberOfAccounts实例变量；

- 修改构造方法，将accounts实例变量初始化为新的ArrayList对象；

- 修改addAccount方法，使用List接口的add()方法；

- 修改getAccount方法，使用List接口的get()方法；

- 修改getNumberOfAccounts方法，使用List接口的size()方法。

6. 修改Bank类：

- 位于包com.mybank.domain中；

- 声明customers静态变量为List<Customer>类型，使用泛型，不再使用numberOfCustomers实例变量；

- 通过静态初始化块，将customers实例变量初始化为新的ArrayList对象；

- 修改静态方法addCustomer()，使用List接口的add()方法；

- 修改静态方法getCustomer()，使用List接口的get()方法；

- 修改静态方法getNumberOfCustomers()，使用List接口的size()方法。

实验代码 ::

容器Bank

```
1 package experiment.six.com.mybank.domain;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 public class Bank {
```

```

7      static List<Customer> customers;
8      static {
9          customers = new ArrayList<>();
10     }
11     public static void addCustomer(String firstName, String lastName){
12         customers.add(new Customer(firstName,lastName));
13     }
14     public static int getNumbersOfCustomers(){
15         return customers.size();
16     }
17     public static Customer getCustomer(int id){
18         try {
19             return customers.get(id);
20         } catch (ArrayIndexOutOfBoundsException e){
21             System.out.println(e);
22         }
23         return null;
24     }
25 }

```

容器*Customer*

```

1  package experiment.six.com.mybank.domain;
2
3  import java.util.ArrayList;
4  import java.util.List;
5
6  public class Customer {
7      String firstName;
8      String lastName;
9      List<Account> accounts;
10     //Account[] accounts;
11     int numbersOfAccounts;
12
13     public Customer(String firstName, String lastName) {
14         this.firstName = firstName;
15         this.lastName = lastName;
16         accounts = new ArrayList<>();
17     }
18     public void addAccount(double amount){
19         addAccount(new Account(amount));
20     }
21     public void addAccount(Account acc){
22         accounts.add(acc);
23     }
24     public Account getAccount(int id){
25         try {

```

```

26         return accounts.get(id);
27     }catch (ArrayIndexOutOfBoundsException e){
28         System.out.println(e.getMessage());
29         return null;
30     }
31 }
32 public int getNumbersOfAccounts() {
33     return accounts.size();
34 }
35 public String getFirstName() {
36     return firstName;
37 }
38
39 public String getLastName() {
40     return lastName;
41 }
42
43 @Override
44 public String toString() {
45     return firstName + "-" + lastName ;
46 }
47 }

```

异常类 *OverdraftException*

```

1 package experiment.six.com.mybank.domain;
2
3 /**
4  * @author liyajun
5  * @date 2022/4/6 14:40
6  */
7 public class OverdraftException extends Exception{
8     private double deficit;
9
10    public OverdraftException(String message,double deficit){
11        super(message);
12        this.deficit = deficit;
13    }
14
15    public double getDeficit() {
16        return deficit;
17    }
18 }

```

测试 ::

```

1 package experiment.six.com.mybank.test;
2
3
4 import experiment.six.com.mybank.domain.*;
5
6
7 public class TestBanking2 {
8     public static void main(String[] args) {
9         Bank.addCustomer("zhang", "san");
10        Customer c1 = Bank.getCustomer(0);
11        c1.addAccount(new SavingsAccount(500,0.017));
12        c1.addAccount(new CheckingAccount(2000, 3000));
13        for(int i = 0; i < Bank.getNumbersOfCustomers(); i++) {
14            Customer cus = Bank.getCustomer(i);
15            System.out.println(cus);
16            for(int j = 0;j < cus.getNumbersOfAccounts();j++) {
17                Account acc = cus.getAccount(j);
18                System.out.println("第" + (j+1) + "个账户");
19                if(acc instanceof SavingsAccount sa){
20                    System.out.printf("计算余额 %.2f\n",sa.accumulateInterest());
21                }
22                else if(acc instanceof CheckingAccount ck){
23                    System.out.printf("当前余额为%.2f\n",ck.getBalance());
24                    System.out.printf("能够透支的钱数 %.2f\n",ck.getOverdraftLimit());
25                    try {
26                        ck.withdraw(6000);
27                    } catch (OverdraftException e) {
28                        System.out.println(e.getMessage()+"赤字为"+e.getDeficit());
29                        e.printStackTrace();
30                    }
31                }
32                else {
33                    //普通账户
34                    try {
35                        acc.withdraw(60000);
36                    } catch (OverdraftException e) {
37                        System.out.println(e.getMessage());
38                        e.printStackTrace();
39                    }
40                    System.out.printf("余额为 %.2f",acc.getBalance());
41                }
42            }
43        }
44    }
45 }
46

```

```
C:\Users\liyaj\jdk\openjdk-17.0.2\bin\java.exe -javaagent:C:\Users\liyaj\AppData\Local\JetBrains\Toolbox\apps\IDEA-U\ch-0\213.7172.25\lib\idea_rt.jar=64712:C:\Users\liyaj\AppData\Local\JetBrains\Toolbox\apps\IDEA-U\ch-0\213.7172.25\bin -Dfile.encoding=UTF-8 -classpath C:\Users\liyaj\IdeaProjects\2022_Java\out\production\2022_Java_experiment.six.com.mybank.test.TestBanking2
zhang-san
第1个账户
计算余额 500.50
第2个账户
当前余额为2000.00
能够透支的钱数 3000.00
透支额度不足赤字为4000.0
experiment.six.com.mybank.domain.OverdraftException: 透支额度不足
    at experiment.six.com.mybank.domain.CheckingAccount.withdraw(CheckingAccount.java:26)
    at experiment.six.com.mybank.test.TestBanking2.main(TestBanking2.java:26)
Process finished with exit code 0
```

总结 ::

学了些了Java的异常类。能够定义、创建并使用异常类。

实验七 多线程技术 .

一、实验目的 ::

1. 掌握线程概念模型和线程生命周期中的状态转换
2. 掌握线程的创建方法与线程同步
3. 掌握线程的同步与互斥
4. 掌握线程之间的通信以及多线程程序设计方法

二、预备知识 ::

1. JDK的安装设置
2. Eclipse/IDEA集成开发环境的安装

三、实验描述 ::

1. 实验类型：设计
2. 实验学时：2学时
3. 实验内容：参照第四项，其中实验内容一必做，实验内容二和三选做

四、实验内容 ::

(1) 检查所使用的计算机系统：

1. 确认是否已安装JDK，并确认系统的环境变量设置
2. 确认是否已安装Eclipse/IDEA集成开发环境

(2) 实验内容一：

设计一个生产者—消费者线程应用：一个字符生产者(Producer) 向一个堆栈 (SyncStack) 中加入字符产品，一个字符消费者 (Consumer) 向同一个堆栈中取出产品。

要求：当多个线程竞争共享资源—堆栈 (MyStack) 时，通过线程同步以保护共享资源；并且通过线程通信使多个线程协调完成工作。例如当堆栈为空时消费者无法取出产品，应先通知生产者加入产品；当堆栈已满时生产者无法继续加入产品，应先通知消费者取出产品。

- SyncTest类：提供程序入口main()方法，负责创建生产者和消费者线程，并且启动这些线程。
- Producer类：生产者线程，通过调用MyStack类的push()方法不断向堆栈中加入产品。
- Consumer类：消费者线程，通过调用MyStack类的pop()方法不断向堆栈中取出产品。
- MyStack类：堆栈，允许从堆栈中取出或加入产品，要保证push()方法、pop()方法和getPoint()方法的同步。

(3) 实验内容二：

继续改造“银行账户管理”项目：

1. 定义一个新的类ConcurrentAccount，代表可以并发访问的账户类。
2. 修改代码：为ConcurrentAccount类的原子操作：deposit()方法和withdraw()方法，分别添加synchronized 标记。
3. 修改代码：使用wait()和notify()/notifyAll()方法

项目中的两个不同线程，分别执行“存钱”和“取钱”任务。其实，这两个线程之间是需要相互通信的，以便协调完成工作。比如：当账户余额是0时，此时不应该再取钱，而是应该通知存钱的线程先去存一些钱。

4. 修改代码：实例变量balance增加volatile关键字修饰

关键字synchronized用于修饰方法或者代码段，所以是块状地对代码进行锁定。volatile关键字则与synchronized不一样，它主要用来修饰变量，可以看做对修饰的变量，进行了读或者写的同步操作。

5. 修改代码：引入重入锁机制，了解Lock类的用法。

(4) 实验内容三：

在学习线程技术知识之后，可以考虑改进猜数字游戏，从而实现多线程猜数字游戏。

要求：用两个线程玩猜数字游戏，第一个线程负责随机给出1-100之间的一个整数，第二个线程负责猜出这个数，要求每当第二个线程给出自己的猜测之后，第一个线程都会提示“猜大了”、“猜小了”或者“猜对了”。猜数之前，要求第二个线程等待第一个线程设置好要猜的数。第一个线程设置好猜测数之后，两个线程还要相互等待，其原则是：第二个线程给出自己的猜测后，等待第一个线程给出的提示；第一个线程给出提示后，等待第二个线程给出猜测，如此进行，直到第二个线程给出正确的猜测后，两个线程进入死亡状态。

生产者消费者 ::

实验代码 ::

Consumer ::

```
1 package experiment.seven.text01;
2
3 /**
4  * @author liyajun
5  * @date 2022/4/18 14:58
6  */
7 public class Consumer extends Thread{
8     private MyStack myStack;
9     public Consumer(MyStack my){
10         this.myStack = my;
11     }
12     @Override
13     public void run(){
14         while(true){
15             myStack.pop();
16         }
17     }
18 }
```

Producer ::

```
1 package experiment.seven.text01;
2
3 /**
4  * @author liyajun
5  * @date 2022/4/18 14:58
6  */
7 public class Producer extends Thread{
8     private MyStack myStack;
9     public Producer(MyStack my){
10         this.myStack = my;
11     }
12     @Override
13     public void run(){
14         while(true){
15             //if(myStack.getPoint() == MyStack.SIZE) this.notifyAll();
16             myStack.push('c');
17         }
18     }
19
20 }
```

```
1 package experiment.seven.text01;
2
3 import java.util.List;
4 import java.util.Stack;
5
6 /**
7  * @author liyajun
8  * @date 2022/4/18 14:58
9  */
10 public class MyStack {
11     static final int SIZE = 10;
12     private volatile Stack<Character> st = new Stack<>();
13     public synchronized void push(char c){
14         if(st.size() > SIZE){
15             System.out.println("容量已满");
16             try {
17                 this.wait();
18                 this.notify();
19             } catch (InterruptedException e) {
20                 e.printStackTrace();
21             }
22         } else {
23             System.out.println(Thread.currentThread().getName() + "生产了"+
24                 this.notify();
25             st.push(c);
26         }
27     }
28     public synchronized void pop(){
29         if(st.size() == 0){
30             System.out.println("容量为空请生产");
31             try {
32                 this.wait();
33                 this.notify();
34             } catch (InterruptedException e) {
35                 e.printStackTrace();
36             }
37         } else {
38             System.out.println(Thread.currentThread().getName() + "取走了"+
39                 this.notify();
40         }
41     }
42     public int getPoint(){
43         return st.size();
44     }
45 }
```

SyncTest

```
1 package experiment.seven.text01;
2
3 /**
4  * @author liyajun
5  * @date 2022/4/18 14:58
6  */
7 public class SyncTest {
8     public static void main(String[] args) {
9         MyStack my = new MyStack();
10        Consumer co = new Consumer(my);
11        Producer pr = new Producer(my);
12        co.start();
13        pr.start();
14    }
15 }
```

测试结果

```
Thread-0取走了c
Thread-0取走了c
容量为空请生产
Thread-1生产了c
Thread-1生产了c
Thread-1生产了c
Thread-1生产了c
Thread-1生产了c
Thread-1生产了c
Thread-1生产了c
Thread-1生产了c
Thread-1生产了c
Thread-1生产了c
容量已满
Thread-0取走了c
Thread-0取走了c
Thread-0取走了c
Thread-0取走了c
Thread-0取走了c
Thread-0取走了c
Thread-0取走了c
Thread-1生产了c
Thread-1生产了c
Thread-1生产了c
Thread-1生产了c
Thread-1生产了c
Thread-1生产了c
Thread-1生产了c
Thread-1生产了c
```

改写银行 ::

实验代码 ::

volatile *and* **synchronized** ::

```
1 package experiment.seven.new_bank.com.domain;
2
3 /**
4  * @author liyajun
5  * @date 2022/4/18 14:03
6  */
7 public class ConcurrentAccount {
8     private volatile double balance = 0.0;
9 }
```

```

10    /**
11     * 存钱
12     * @param amt
13     */
14    public synchronized void deposit(double amt){
15        this.notifyAll();
16        balance += amt;
17        System.out.println(Thread.currentThread().getName()+"存入"+amt+"元");
18    }
19
20    /**
21     * 取钱
22     * @param amt
23     */
24    public synchronized void withdraw(double amt){
25        if(balance == 0) {
26            System.out.println(Thread.currentThread().getName()+"余额不足");
27            try {
28                this.wait();
29            } catch (InterruptedException e) {
30                e.printStackTrace();
31            }
32        }
33        if(balance >= amt){
34            balance -= amt;
35            System.out.println(Thread.currentThread().getName()+"取出"+amt+"元");
36        } else {
37            System.out.println(Thread.currentThread().getName()+"余额不足");
38        }
39    }
40
41    /**
42     * 获取余额
43     */
44    public void getBalance(){
45        System.out.println("账户余额为" + balance+"元");
46    }
47 }
48

```

使用Lock

```

1 package experiment.seven.new_bank.com.domain;
2
3 import java.util.concurrent.locks.Lock;
4 import java.util.concurrent.locks.ReentrantLock;
5

```

```

6  /**
7   * @author liyajun
8   * @date 2022/4/18 14:03
9   */
10 public class ConcurrentAccount {
11     private volatile double balance = 0.0;
12     private Lock lock = new ReentrantLock();
13
14     /**
15      * 存钱
16      * @param amt
17      */
18     public void deposit(double amt){
19         lock.lock();
20         balance += amt;
21         System.out.println(Thread.currentThread().getName()+"存入"+amt+"元");
22         lock.unlock();
23     }
24
25     /**
26      * 取钱
27      * @param amt
28      */
29     public void withdraw(double amt){
30         lock.lock();
31         if(balance >= amt){
32             balance -= amt;
33             System.out.println(Thread.currentThread().getName()+"取出"+amt+"元");
34         } else {
35             System.out.println(Thread.currentThread().getName()+"余额不足");
36         }
37         lock.unlock();
38     }
39
40     /**
41      * 获取余额
42      */
43     public void getBalance(){
44         System.out.println("账户余额为" + balance+"元");
45     }
46 }

```

test

```

1 package experiment.seven.new_bank.com.test;
2
3 import experiment.seven.new_bank.com.domain.ConcurrentAccount;

```

```
4
5 /**
6  * @author liyajun
7  * @date 2022/4/18 14:09
8  */
9 public class TestBanking3 {
10     public static void main(String[] args) {
11         ConcurrentAccount ca = new ConcurrentAccount();
12         new Thread(() -> {
13             while (true){
14                 ca.deposit(114);
15                 try {
16                     Thread.sleep(300);
17                 } catch (InterruptedException e) {
18                     e.printStackTrace();
19                 }
20                 ca.getBalance();
21             }
22         }).start();
23         new Thread(() -> {
24             while (true){
25                 ca.withdraw(114);
26                 try {
27                     Thread.sleep(300);
28                 } catch (InterruptedException e) {
29                     e.printStackTrace();
30                 }
31                 ca.getBalance();
32             }
33         }).start();
34     }
35 }
```

运行结果 ::

```
Thread-0存入114.0元
Thread-1取出114.0元
账户余额为0.0元
Thread-0存入114.0元
账户余额为114.0元
Thread-1取出114.0元
账户余额为0.0元
账户余额为0.0元
Thread-1余额不足
Thread-0存入114.0元
Thread-1取出114.0元
账户余额为0.0元
账户余额为0.0元
Thread-1余额不足
Thread-0存入114.0元
Thread-1取出114.0元
账户余额为0.0元
Thread-0存入114.0元
账户余额为0.0元
Thread-1取出114.0元
账户余额为0.0元
账户余额为0.0元
Thread-0存入114.0元
Thread-1取出114.0元
账户余额为0.0元
账户余额为0.0元
Thread-0存入114.0元
```

改进版猜数字

代码

GenNum

生成数字

```
1 package experiment.seven.text03;
2
3 import java.util.Random;
4
5 /**
6  * @author liyajun
7  * @date 2022/4/18 16:02
8  */
9 public class GenNum implements Runnable{
10     int num;
11     @Override
12     public synchronized void run(){
```



```

13         Random ran = new Random();
14         this.num = ran.nextInt(100);
15     }
16     public int getNum(){
17         return num;
18     }
19     public void setNum(int num){
20         this.num = num;
21     }
22 }

```

猜数字

```

1 package experiment.seven.text03;
2
3 /**
4  * @author liyajun
5  * @date 2022/4/18 15:56
6  * 出数字的线程为守护线程
7  */
8 public class Num {
9
10     public static void main(String[] args) {
11         GenNum n1 = new GenNum();
12         Thread t1 = new Thread(n1);
13         t1.run();
14         t1.setDaemon(true);
15         try {
16             Thread.sleep(100);
17             GenNum n2 = new GenNum();
18             Thread t2 = new Thread(n2);
19             while(true){
20                 t2.interrupt();
21                 t2.run();
22                 Thread.sleep(100);
23                 t1.interrupt();
24                 if(n1.getNum() == n2.getNum()){
25                     System.out.println("猜的数字为"+n2.getNum());
26                     System.out.println("猜对了!");
27                     break;
28                 } else if(n1.getNum() > n2.getNum()){
29                     System.out.println("猜的数字为"+n2.getNum());
30                     System.out.println("猜小了");
31                 } else {
32                     System.out.println("猜的数字为"+n2.getNum());
33                     System.out.println("猜大了");
34                 }

```

```

35         }
36     } catch (InterruptedException e) {
37         e.printStackTrace();
38     }
39
40 }
41 }

```

测试结果 ::

```

C:\Users\liyaj\jdk\openjdk-17.0.2\bin>java.exe -javaagent:C:\Users\liyaj\AppData\Local\JetBrains\Toolbox\apps\IDEA-U\ch-0\213.7172.25\lib\idea_lic
.jar=52053:C:\Users\liyaj\AppData\Local\JetBrains\Toolbox\apps\IDEA-U\ch-0\213.7172.25\bin -Dfile.encoding=UTF-8 -classpath C:\Users\liyaj\IdeaProjects\2022_Java\out\pro
experiment.seven.text03.Num
猜的数字为90
猜大了
猜的数字为73
猜小了
猜的数字为14
猜小了
猜的数字为39
猜小了
猜的数字为88
猜大了
猜的数字为95
猜大了
猜的数字为95
猜大了
猜的数字为69
猜小了
猜的数字为16
猜小了
猜的数字为76
猜小了
猜的数字为3
猜小了
猜的数字为81
猜大了
猜的数字为57
猜小了
猜的数字为85
猜大了
猜的数字为41
猜小了
猜的数字为80
猜对了!
Process finished with exit code 0

```

尾 ::

单纯的随机猜靠运气，没有给另一个线程加上二分，可以加上二分提速。

总结 ::

学习Java的多线程技术 能够使用多线程处理一些问题，也理解多线程里面的一些东西和关键字

实验八 银行账户管理 — 输入输出操作 ::

银行账户修改 ::

一、实验目的 ::

1. 熟练运用Java I/O API读取/写入数据文件
2. 熟练掌握Java 面向对象的编程

二、预备知识 ::

1. JDK的安装设置

2. Eclipse/IDEA集成开发环境的安装
3. 掌握Java语言的面向对象特性（封装性、继承性、多态性）
4. 掌握Java应用编程关于输入输出流等相关知识

三、实验描述 ::

1. 实验类型：设计
2. 实验学时：2学时
3. 实验内容：1项（参照第四项）

四、实验内容 ::

(1) 检查所使用的计算机系统：

1. 确认是否已安装JDK，并确认系统的环境变量设置；
2. 确认是否已安装Eclipse/IDEA集成开发环境。

(2) 实验分析：

寻找客户数据文件的规律，例如：**test.dat**

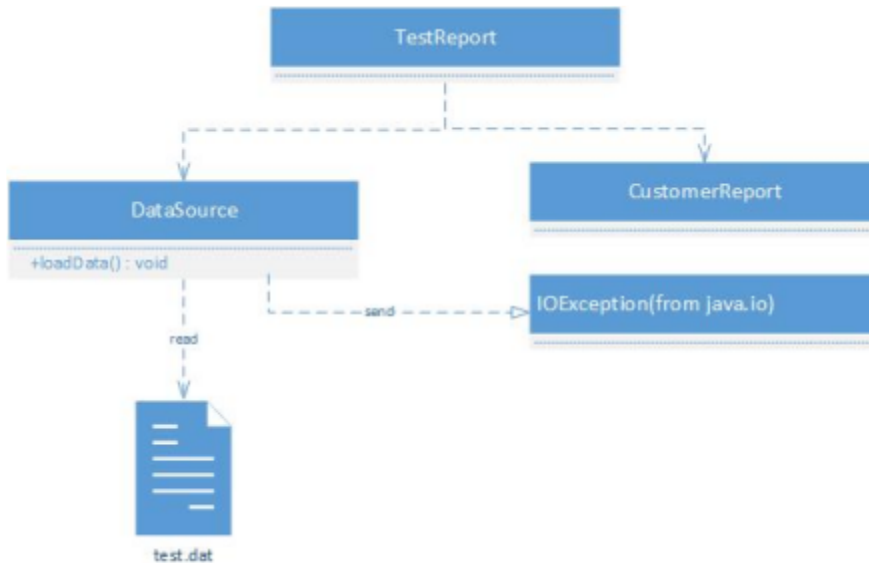
```
1 4
2
3 Jane    Simms    2
4
5 S       500.00    0.05
6
7 C       200.00    400.00
8
9 Owen    Bryant   1
10
11 C       200.00    0.00
12
13 Tim     Soley     2
14
15 S       1500.00   0.05
16
17 C       200.00    0.00
18
19 Maria   Soley     1
20
21 S       150.00    0.05
22
```

可以很容易地区分出客户数目，每个客户的姓名、以及拥有的账户数目和类型，还有每个账户的初始余额、利率、以及允许透支的额度等。客户数据彼此间可以使用制表符分隔。

Java 5添加了Scanner类，位于util包中。借助Scanner类的nextXXX()和hasNextXXX()方法，可以任意地对字符串和基本类型的数据进行分析。

关于报表输出，可以使用PrintWriter类、或者BufferedWriter类，将用户数据写入目的地。此处将PrintWriter和FileWriter流连接在一起，主要是看中PrintWriter类的println()方法可以按行输出。

(3) 实验内容：继续修改项目：BankProject



1. 读取数据文件

- 创建资源文件test.dat，该文件存放用户账户的基本信息；
- 创建DataSource类：位于com.mybank.data包中；
- 添加公有方法loadData()，要求使用Scanner类从test.dat中格式化读取Bank客户对象和每个客户的账户对象。

2. 创建CustomerReport类：

- 位于包com.mybank.report中；
- 添加公有方法generateReport，要求利用PrintWriter和FileWriter对象格式化输出银行的客户及账户信息。

3. 创建TestReport类：

- 位于包com.mybank.test中；
- 要求分别初始化DataSource、CustomerReport类实例对象，然后分别调用loadData()方法和generateReport()方法实现格式化读取数据和输出数据。

实现代码 ::

DataSource

```
1 package experiment.eight.com.mybank.data;
2
3 import experiment.eight.com.mybank.domain.Bank;
4 import experiment.eight.com.mybank.domain.CheckingAccount;
5 import experiment.eight.com.mybank.domain.Customer;
6 import experiment.eight.com.mybank.domain.SavingsAccount;
7
8 import java.io.File;
9 import java.io.FileNotFoundException;
10 import java.util.Scanner;
11
12 /**
13  * @author liyajun
14  * @date 2022/5/2 13:16
15  */
16 public class DataSource {
17     public void loadData(File file){
18         try {
19             Scanner sc = new Scanner(file);
20             //get the count of new add acc
21             int add_count_num = sc.nextInt();
22             for(int i = 0;i < add_count_num;i++){
23                 String firstName = sc.next();
24                 String lastName = sc.next();
25                 Bank.addCustomer(firstName, lastName);
26                 Customer cus_cur = Bank.getCustomer(i);
27                 // cus account count
28                 int count_cus_acc = sc.nextInt();
29                 for (int j = 0; j < count_cus_acc; j++) {
30                     String accType = sc.next();
31                     if(accType.startsWith("S")){
32                         double initBalance = sc.nextDouble();
33                         double growRate = sc.nextDouble();
34                         cus_cur.addAccount(new SavingsAccount(initBalance, growRate));
35                     }
36                     else {
37                         double initBalance = sc.nextDouble();
38                         double overdraft = sc.nextDouble();
39                         cus_cur.addAccount(new CheckingAccount(initBalance, overdraft));
40                     }
41                 }
42             }
43         } catch (FileNotFoundException e) {
```

```

44         e.printStackTrace();
45     }
46 }
47 }
48

```

CustomerReport

```

1 package experiment.eight.com.mybank.report;
2
3 import experiment.eight.com.mybank.domain.Bank;
4 import experiment.eight.com.mybank.domain.Customer;
5
6 import java.io.File;
7 import java.io.FileWriter;
8 import java.io.IOException;
9 import java.io.PrintWriter;
10
11 /**
12  * @author liyajun
13  * @date 2022/5/2 13:31
14  */
15 public class CustomerReport {
16     public void generateReport(){
17         File file = new File("src\\experiment\\eight\\com\\mybank\\TestData");
18         // System.out.println("start");
19         try (PrintWriter pt = new PrintWriter(new FileWriter(file))) {
20             pt.println("_____ split line _____");
21             for(int i = 0; i < Bank.getNumbersOfCustomers(); i++){
22                 Customer customer = Bank.getCustomer(i);
23                 pt.println(customer.toString() + " numbers of account " + customer.getNumbersOfAccount());
24             }
25             pt.println("_____ split line _____");
26         } catch (IOException e) {
27             e.printStackTrace();
28         }
29     }
30 }
31 }
32

```

TestReport

```

1 package experiment.eight.com.mybank.test;
2
3 import experiment.eight.com.mybank.data.DataSource;

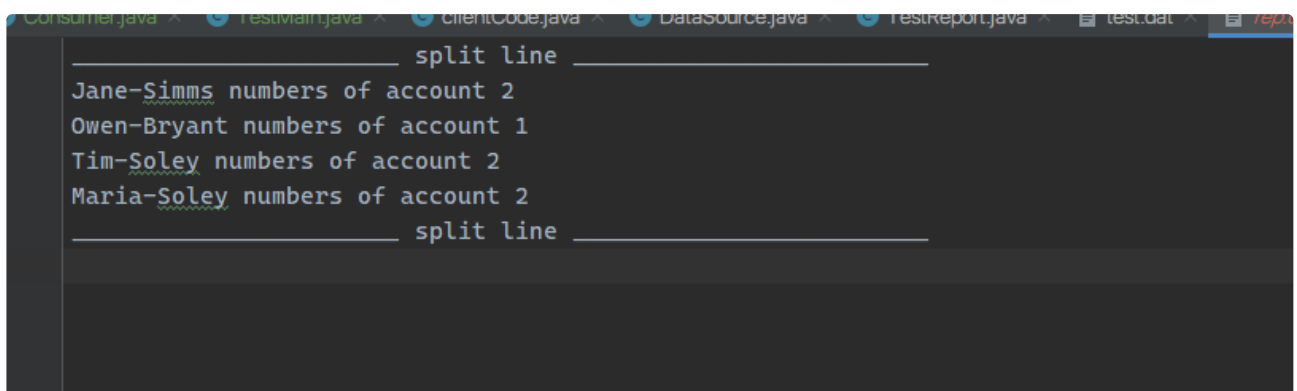
```

```

4 import experiment.eight.com.mybank.report.CustomerReport;
5
6 import java.io.File;
7
8 /**
9  * @author liyajun
10  * @date 2022/5/2 13:38
11  */
12 public class TestReport {
13     public static void main(String[] args) {
14         File file = new File("src\\experiment\\eight\\com\\mybank\\TestData");
15         DataSource ds = new DataSource();
16         ds.loadData(file);
17         CustomerReport cs = new CustomerReport();
18         cs.generateReport();
19     }
20 }

```

测试结果



```

split line
Jane-Simms numbers of account 2
Owen-Bryant numbers of account 1
Tim-Soley numbers of account 2
Maria-Soley numbers of account 2
split line

```

总结

学习了Java的文件读写以及一些文件异常操作

实验九 与服务器猜数字

一、实验目的

1. 了解Java 语言的网络通信功能，以及Socket通信的概念
2. 掌握Socket通信的编程方法

二、准备工作

1. JDK的安装设置
2. Eclipse/IDEA集成开发环境的安装

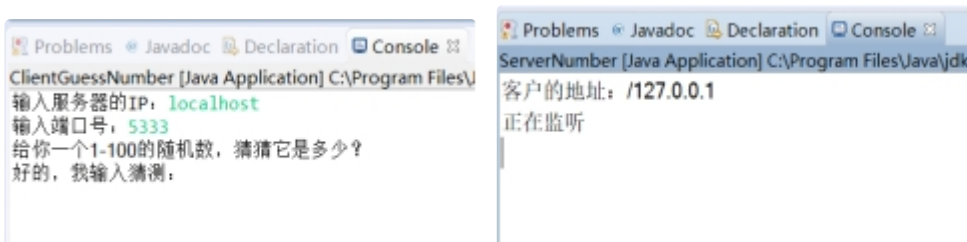
三、实验描述 ::

1. 实验类型：设计
2. 实验学时：2学时
3. 实验内容：

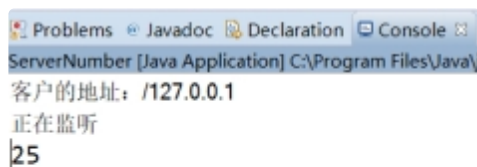
在学习Java网络编程之后，可以继续改进猜数字游戏，从而实现与服务器玩猜数字游戏。客户端和服务端建立套接字连接后，服务器向客户发送一个1~100的随机数，用户将自己的猜测发送给服务器，服务器向用户发送有关信息：“猜大了”，“猜小了”或“猜对了”，直到猜对为止。

4. 程序运行过程简单演示：

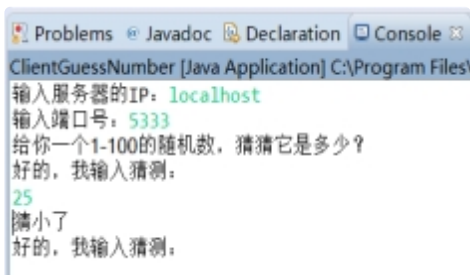
- 首先，把服务器端程序运行起来。
- 然后，运行客户端程序，要求输入服务器端的IP 地址和开放的端口号。
- 随后，服务器端发送过来一个 1~100 之间的随机数，让客户端猜测。



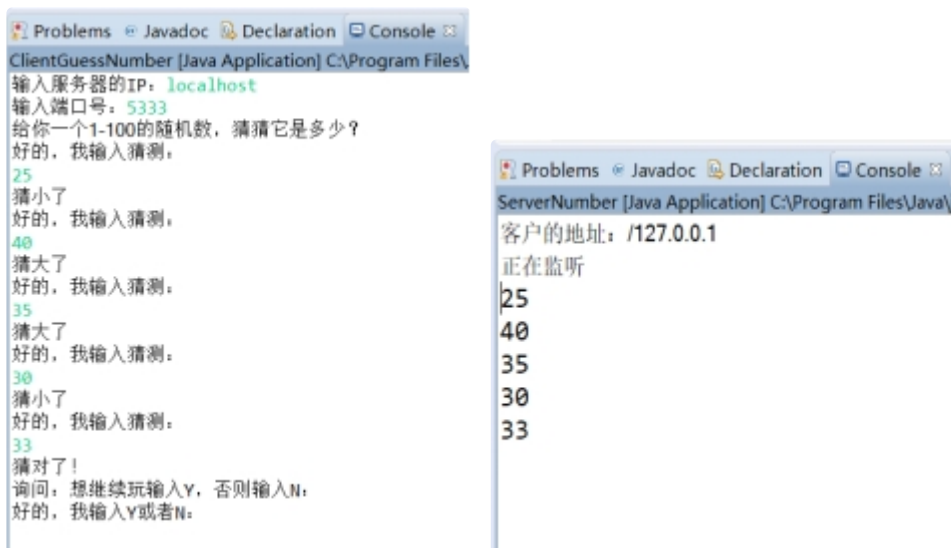
- 客户端输入猜测值：25。



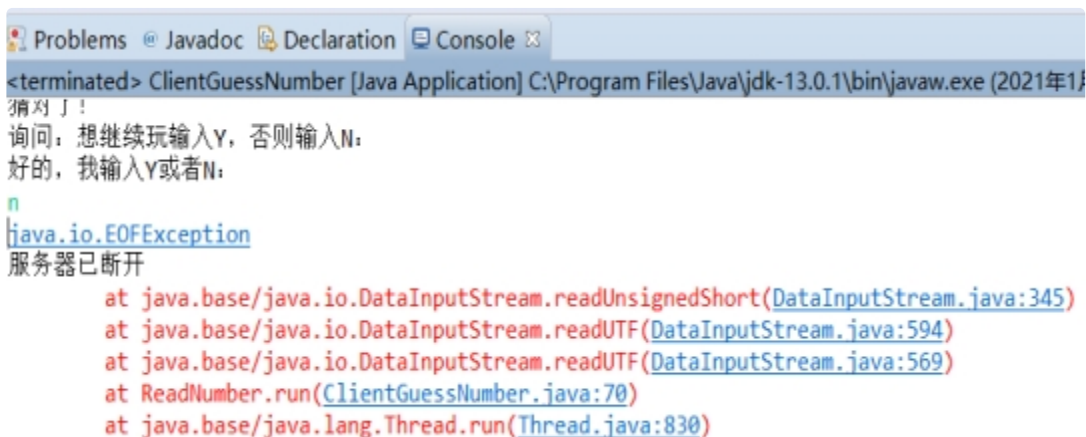
- 服务器端收到后，先做判断，然后给出结果：“猜小了”。



- 重复上述过程，直至猜对为止。服务器会继续询问是否继续猜数字。



- 如果客户端输入了“N”或者“n”，服务器会断开连接。



实现代码 ::

clientCode

```

1 package experiment.nine;
2
3 import java.io.*;
4 import java.net.InetAddress;
5 import java.net.InetSocketAddress;
6 import java.net.Socket;
7 import java.util.Scanner;
8
9 /**
10  * @author liyajun
11  * @date 2022/5/11 14:33
12  */
13 public class clientCode {
14     public static void main(String[] args) {
15         Scanner sc = new Scanner(System.in);

```

```

16     Socket mySocket = null;
17     DataInputStream inData = null;
18     DataOutputStream outData = null;
19     ReadNumber readNumber = null;
20     Thread thread;
21     try {
22         mySocket = new Socket();
23         readNumber = new ReadNumber();
24         thread = new Thread(readNumber);
25         System.out.print("输入服务器的IP: ");
26         String ip = sc.nextLine();
27         System.out.print("输入端口号: ");
28         int port = sc.nextInt();
29
30         if (!mySocket.isConnected()) {
31             InetAddress address = InetAddress.getByName(ip);
32             InetSocketAddress socketAddress = new InetSocketAddress(
33                 address, port);
34             mySocket.connect(socketAddress);
35             InputStream in = mySocket.getInputStream();
36             OutputStream out = mySocket.getOutputStream();
37             inData = new DataInputStream(in);
38             outData = new DataOutputStream(out);
39             readNumber.setDataInputStream(inData);
40             readNumber.setDataOutputStream(outData);
41             thread.start();
42         }
43     } catch (Exception e) {
44         System.out.println("服务器已断开");
45     }
46 }
47 }
48
49 class ReadNumber implements Runnable {
50     Scanner sc = new Scanner(System.in);
51     DataInputStream in;
52     DataOutputStream out;
53
54     public void setDataInputStream(DataInputStream in) {
55         this.in = in;
56     }
57
58     public void setDataOutputStream(DataOutputStream out) {
59         this.out = out;
60     }
61
62     @Override

```

```

63     public void run() {
64         try {
65             out.writeUTF("Y");
66             while (true) {
67                 String str = in.readUTF();
68                 System.out.println(str);
69                 if (!str.startsWith("询问")) {
70                     if (str.startsWith("猜对了")) continue;
71                     System.out.println("好的，我输入猜测：");
72                     int myGuess = sc.nextInt();
73                     String enter = sc.nextLine();
74                     out.writeInt(myGuess);
75                 } else {
76                     System.out.println("好的，我输入Y或者N：");
77                     String myAnswer = sc.nextLine();
78                     out.writeUTF(myAnswer);
79                 }
80             }
81         } catch (IOException e) {
82             e.printStackTrace();
83             System.out.println("服务器已断开");
84         }
85     }
86 }

```

serverCode

```

1 package experiment.nine;
2
3 import java.io.*;
4 import java.net.ServerSocket;
5 import java.net.Socket;
6 import java.util.Random;
7
8 /**
9  * @author liyajun
10  * @date 2022/5/11 14:33
11  */
12 public class serverCode {
13     public static void main(String[] args) {
14         ServerSocket server = null;
15         Socket you = null;
16         while (true) {
17             try {
18                 server = new ServerSocket(5333);

```

```

19         } catch (IOException e) {
20             System.out.println("正在监听");
21         }
22         try {
23             you = server.accept();
24             System.out.println("客户的地址: " + you.getInetAddress());
25         } catch (IOException e) {
26             e.printStackTrace();
27         }
28         if (you != null) {
29             new ServerThread(you).start();
30         }
31     }
32 }
33 }
34
35 class ServerThread extends Thread {
36     Socket socket;
37     DataInputStream in = null;
38     DataOutputStream out = null;
39
40     ServerThread(Socket t) {
41         socket = t;
42         try {
43             out = new DataOutputStream(socket.getOutputStream());
44             in = new DataInputStream(socket.getInputStream());
45         } catch (IOException e) {
46             e.printStackTrace();
47         }
48     }
49
50     public void run() {
51         try {
52             while (true) {
53                 String str = in.readUTF();
54                 boolean boo = str.startsWith("Y") || str.startsWith("y");
55                 if (boo) {
56                     out.writeUTF("给你一个1-100的随机数, 猜猜它是多少? ");
57                     Random ran = new Random();
58                     int realNumber = ran.nextInt(100) + 1;
59                     handleClientGuess(realNumber);
60                     out.writeUTF("询问: 想继续玩输入Y, 否则输入N: ");
61                 }
62             }
63         } catch (IOException e) {
64             e.printStackTrace();
65         }

```

```

66     }
67
68     public void handleClientGuess(int realNumber) {
69         while (true) {
70             try {
71                 int clientGuess = in.readInt();
72                 System.out.println(clientGuess);
73                 if (clientGuess > realNumber)
74                     out.writeUTF("猜大了");
75                 else if (clientGuess < realNumber)
76                     out.writeUTF("猜小了");
77                 else {
78                     out.writeUTF("猜对了! ");
79                     break;
80                 }
81             } catch (IOException e) {
82                 System.out.println("客户离开");
83                 return;
84             }
85         }
86     }
87 }

```

总结 ::

了解了 *Java* 语言的网络通信功能，以及 *Socket* 通信的概念。学习了 *Socket* 通信的编程方法

实验十 JDBC数据库操作 .

一、实验目的 ::

1. 了解JDBC 体系结构中的层以及驱动程序的种类
2. 掌握JDBC API的类和接口
3. 掌握创建JDBC应用程序的方法

二、准备工作 ::

1. JDK的安装设置
2. Eclipse/IDEA集成开发环境的安装

三、实验描述 ::

1. 实验类型：设计
2. 实验学时：2学时

3. 实验内容：创建使用PreparedStatement对象的数据库操作应用程序

4. 实验准备：

- (1) 安装MySQL (端口号、root和密码)
- (2) 安装Navicat (数据库的可视化工具, 可选)
- (3) 创建数据库连接、数据库和表 (student)
- (4) 在Java Project中引入MySQL jar包
- (5) 编写代码 (使用PreparedStatement对象)

5. 实验编程要求：

访问MySQL (或者SQLServer) 数据库, 使用 JDBC API构建应用程序, 执行以下各种数据库操作：查询表、在表中插入行、在表中更新行、从表中删除行。

- (1) 装入驱动程序到驱动程序管理器DriverManager
- (2) 使用正确的URL连接到数据库
- (3) 要求使用PreparedStatement对象创建并执行预处理语句, 如果是查询操作, 要求显示ResultSet结果集数据
- (4) 处理 SQL 异常

实验代码 ::

prepareStatement

```
1 package experiment.ten;
2
3 import java.sql.*;
4
5 /**
6  * @author liyajun
7  * @date 2022/5/30 14:46
8  */
9 public class preparedStatement {
10     public static void main(String[] args) throws SQLException {
11         String driver="com.mysql.cj.jdbc.Driver";
12         String URL="jdbc:mysql://localhost:3306/spdb?serverTimezone=UTC&useUnicode=true&characterEncoding=utf8";
13         Connection conn = null;
14         String name = "root";
15         String password = "root";
16
17         try {
```

```

18         Class.forName(driver); //loading
19
20     } catch (ClassNotFoundException e) {
21         throw new RuntimeException(e);
22     }
23
24     try {
25         conn = DriverManager.getConnection(URL,name,password); //get co
26     } catch (SQLException e) {
27         throw new RuntimeException(e);
28     }
29
30     query(conn);
31     insert(conn);
32     query(conn);
33     update(conn);
34     query(conn);
35     delete(conn);
36     query(conn);
37
38 }
39 static void query(Connection conn) throws SQLException {
40     String sql = "select * from account";
41     PreparedStatement pre = conn.prepareStatement(sql);
42     ResultSet res = pre.executeQuery();
43     while(res.next()){
44         String id = res.getString("id");
45         String na = res.getString("name");
46         String money = res.getString("money");
47         System.out.println(id + " " + na + " " + money);
48     }
49 }
50 static void insert(Connection conn) throws SQLException {
51     String sql = "insert into account values(22,33,44)";
52     PreparedStatement pre = conn.prepareStatement(sql);
53     int count = pre.executeUpdate();
54     System.out.println("插入数据条数" + count);
55 }
56 static void update(Connection conn) throws SQLException {
57     String sql = "update account set money = 114514 where id = 1";
58     PreparedStatement pre = conn.prepareStatement(sql);
59     int count = pre.executeUpdate();
60     System.out.println("修改数据条数" + count);
61 }
62 static void delete(Connection conn) throws SQLException {
63     String sql = "delete from account where id = 22";
64     PreparedStatement pre = conn.prepareStatement(sql);

```

```
65         int count = pre.executeUpdate();
66         System.out.println("删除" + count);
67
68     }
69 }
```

总结

学习了*jdbc*的使用方法，能够使用*jdbc*完成一些简单的操作