面向对象学习笔记之测试代码

一、编程题

1、判断 101-200 之间有多少个素数，并输出所有素数。

分析：

1.1 把 101-200 之间所有的数都遍历一遍。

1.2 把 101-200 之间的每一个数(i)都和 2 到根号 i 之间的每一个数相除，如果能被其中的任何一个数整除，就不是素数，如果不能被 2 到根号 i 之间的所有数整除，那么这个数就是素数。

程序如下：

```
public class PrimeTest {
    public static void main (String[] args) {
        //外层for循环对101-200之间所有的数字进行遍历
        for (int i = 101; i < 200; i++) {
            //内层for循环遍历能被2到根号i之间整除的数字
            for (int j = 2; j < Math.sqrt(i); j++) {
                if (i % j == 0) { //如果一个数能被其他数整除，则不是素数，
                    break;          //终止内层for循环
                }else  if(j > Math.sqrt(i)-1){
                //如果一个数从2到根号i之间所有的数都不能被整除，他就是素数，
                    System.out.print(i+"\t"); //打印该素数
                }else {
                    continue;  //当没有遍历到根号i时，继续内层for循环。
                }
            }//end for inner
        }//end for out
    }//end main
}//end class
```

结果是：

```
101 103 107 109 113 127 131 137 139 149 151
157 163 167 173 179 181 191 193 197 199
```

2、随机数猜数小游戏，用函数生成一个 1-100 之间的随机数，然后从键盘输入一个数与该数进行比较，如果大于随机数就提示“猜大了，请往小了猜”，如果小于随机数就提示 “猜小了，请往大了猜”，如果刚好和随机数相等，则提示“恭喜你，猜对了”

程序如下：

```
class Test {
    public static void main(String[] args) {
        int random = (int)(Math.random()*100 + 1);//1-num 之间随机数
        Scanner sc = new Scanner(System.in);  //创建键盘输入对象
        int number = sc.nextInt();                  //保存键盘录入数字
        while (number != random) {                  //当输入的数字和随机数不相等时
            if (number > random) {                  //判断输入数据是否大于随机数
                System.out.println("猜大了，请往小了猜");
            } else {
```

```
                System.out.println("猜小了，请往大了猜");
            }
            number = sc.nextInt();
        }
        System.out.println("恭喜你，猜对了");
    }
}
```

3、用代码实现成员内部类的使用

分析：成员内部类的使用格式：

外部类名.内部类名 对象名 = new 外部类名().new 内部类名();

程序如下：

```
class Outer {   //外部类
    private int num = 10;     //外部类成员变量
    class Inner {    //成员内部类
        private int num2 = 20;   //内部类成员变量
        public void show(){
            System.out.println(num);
            System.out.println(num2);
        }
    }     //end class Inner
}     //end class Outer

class InnerClassDemo3 {
    public static void main(String[] args) {
        // 外部类名.内部类名 对象名 = new 外部类名().new 内部类名();
        Outer.Inner oi = new Outer().new Inner();
        oi.show();//成员内部类对象名 . 成员内部类方法名
    }
}
```

4、多态编程练习

动物园里有很多种动物：比如说，狗，猫等。狗有姓名和年龄，猫也有姓名和年龄。狗有跑步的方法，猫也有跑步的方法。而且都仅仅是跑步。狗有吃饭的方法，狗会看家，猫也有吃饭的方法，猫会钓鱼。只不过，狗吃骨头，猫吃鱼。请用所学知识，对这个问题进行解决。

分析：

父类 Animal：成员变量，成员方法

子类 Dog 和 Cat：重写父类的方法，特有的方法

测试类：普通调用。

```
//动物类
public class Animal {
    private String name;  //成员变量
    private int age;
```

```
    public Animal() {     //构造函数
        super();
    }
    public Animal(String name, int age) {
        super();
        this.name = name;
        this.age = age;
    }
    public String getName() {    //成员函数
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public void run(){
        System.out.println("我会跑步！！！");
    }
    public void eat(){
        System.out.println("吃饭");
    }
}
//狗类
public class Dog extends Animal {
    public Dog() {    //构造函数
    }
    public Dog(String name, int age) {
        super(name, age);
    }
    @Override  //重写父类方法
    public void eat() {
        System.out.println("狗吃骨头");
    }
    public void lookHome(){   //特有的方法
        System.out.println("狗看家");
    }
}
//猫类
public class Cat extends Animal {
```

```
    public Cat() {     ////构造函数
    }
    public Cat(String name, int age) {
        super(name, age);
    }
    @Override  //重写父类方法
    public void eat() {
        System.out.println("猫吃鱼");
    }
    public void playGame(){   //猫类特有方法
        System.out.println("猫钓鱼");
    }
}
//测试类
public class AnimalTest {
    public static void main(String[] args) {
        System.out.println("------多态方式创建dog对象-----");
        Animal an = new Dog("东东",6);
        System.out.println(an.getName()+"--"+an.getAge());
        an.eat();//父类的方法
        //an.lookHome(); Dog类特有方法,父类无法访问。

        System.out.println("-----向下转型-----");
        Dog dog = (Dog)an;//把父类引用强转为子类类型。
        dog.eat();//自己的重写父类的方法
        dog.lookHome();//调用自己特有的方法

        System.out.println("-----多态方法创建Cat对象-----");
        an = new Cat("云云", 8);
        System.out.println(an.getName()+"--"+an.getAge());
        an.eat();//父类的方法
        //an.playGame();//父类引用无法访问子类特有的方法

        System.out.println("-----向下转型-----");
        Cat cat = (Cat)an;//把父类引用强制转换成子类类型
        cat.eat();//自己重写父类的方法
        cat.playGame();//自己特有的方法
    }
}
输出结果:
------多态方式创建dog对象-----
东东--6
狗吃骨头
-----向下转型-----
```

狗吃骨头
狗看家
-----多态方法创建Cat对象-----
云云--8
猫吃鱼
-----向下转型-----
猫吃鱼
猫钓鱼

5、抽象类编程题
对员工进行建模，员工包含 3 个属性：姓名、工号以及工资。经理也是员工，除了含有员工的属性外，另为还有一个奖金属性。请使用继承的思想设计出员工类和经理类。要求类中提供必要的方法进行属性访问。
分析：
父类：员工类(抽象)：Emplyee
成员变量：姓名、年龄、工资
成员函数：getXXX、setXXX
抽象成员函数：work();
子类：经理类：Manager
成员变量：奖金
成员方法：work();
子类：程序袁类：Programer
成员方法：work();

```
//员工类
public abstract class Emplyee {
    //成员变量
    private String name;
    private int age;
    private String salary;
    //构造方法
    public Emplyee() {
        super();
    }
    public Emplyee(String name, int age, String salary) {
        super();
        this.name = name;
        this.age = age;
        this.salary = salary;
    }
    //设置获取方法
    public String getName() {
        return name;
    }
    public void setName(String name) {
```

```
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public String getSalary() {
        return salary;
    }
    public void setSalary(String salary) {
        this.salary = salary;
    }
    //抽象公共方法
    public abstract void work();
}

//项目经理类
public class Manager extends Emplyee {
    //特有变量
    private String bouns;//奖金
    //构造函数
    public Manager() {
    }
    public Manager(String name, int age, String salary, String bouns) {
        super(name, age, salary);
        this.bouns = bouns;
    }
    //设置、获取
    public String getBouns() {
        return bouns;
    }
    public void setBouns(String bouns) {
        this.bouns = bouns;
    }
    @Override//重写父类的方法
    public void work() {
        System.out.println("我是项目经理，我的工作是管理项目的进度");
    }
}
//程序猿类
public class Programer extends Emplyee {
    //构造方法
```

```
    public Programer() {
    }
    public Programer(String name, int age, String salary) {
        super(name, age, salary);
    }

    @Override  //重写父类方法
    public void work() {
        System.out.println("我是程序猿，我的工作是写代码");
    }
}

//测试类
public class Test {
    public static void main(String[] args) {
        Programer p = new Programer("小强", 20, "8k");
        System.out.println("我叫"+p.getName()+",我"+p.getAge()+"岁"+",
            我工资"+p.getSalary());
        p.work();
        System.out.println("----------------------------");
        Manager m = new Manager("斯洛登", 30, "20k","100k");
        System.out.println("我叫"+m.getName()+",我"+m.getAge()+"岁"+",
            我工资"+m.getSalary()+",我的年终奖金是"+m.getBouns());
        m.work();
    }
}
```

输出结果是：

```
我叫小强,我20岁,我工资8k
我是程序猿，我的工作是写代码
----------------------------
我叫斯洛登,我30岁,我工资20k,我的年终奖金是100k
我是项目经理，我的工作是管理项目的进度
```

6、接口编程题

有乒乓球运动员和足球运动员，乒乓球教练和足球教练。为了出国交流，跟乒乓球相关的人员都需要学习英语。请用所学知识用代码实现这个案例中有哪些抽象类，哪些接口，哪些具体类

分析：

接口：英语接口

方法：学习英语 studyEnglish();

抽象类：人

属性 name

方法 eat(抽象方法)

(抽象类)运动员： extends 人:

```
        方法：study(抽象方法)
    (具体类)乒乓球运动员 extends 运动员 implements 英语接口
    (具体类)足球运动员 extends 运动员
    (抽象类)教练 extends 人:
        方法：teach(抽象方法)
    (具体类)乒乓球教练 extends 教练 implements 英语接口
    (具体类)足球教练 extends 教练
//英语接口
interface English {
    public abstract void studyEnglish();//学英语
}

//人：抽象类
abstract class Person {
    //成员变量
    private String name;
    //构造方法
    public Person(){}
    public Person(String name) {
        this.name = name;
    }
    //成员方法
    public void setName(String name){
        this.name = name;
    }
    public String getName(){
        return name;
    }
    //抽象方法:eat()
    public abstract void eat();
}

//(抽象类)运动员 extends 人:
abstract class Player extends Person {
    //构造方法
    public Player(){}
    public Player(String name){
        super(name);
    }
    //方法：study(抽象方法)
    public abstract void study();
}

//(抽象类)教练 extends 人:
```

```
abstract class Coacher extends Person {
	//构造方法
	public Coacher(){}
	public Coacher(String name){
		super(name);
	}
	//方法：teach(抽象方法)
	public abstract void teach();
}

//(具体类)乒乓球运动员  extends  运动员  implements  英语接口
class PingPongPlayer extends Player implements English {
	//构造方法
	public PingPongPlayer(){}
	public PingPongPlayer(String name){
		super(name);
	}
	//重写所有的抽象方法
	public void study(){
		System.out.println("学习扣球");
	}
	public void eat(){
		System.out.println("吃鱼");
	}
	public void studyEnglish(){
		System.out.println("学英语");
	}
}

//(具体类)足球运动员  extends  运动员
class FootballPlayer extends Player {
	//构造方法
	public FootballPlayer(){}
	public FootballPlayer(String name){
		super(name);
	}
	//重写所有的抽象方法
	public void study(){
		System.out.println("学习踢球");
	}
	public void eat(){
		System.out.println("吃馒头");
	}
}
```

```
//(具体类)乒乓球教练 extends 教练 implements 英语接口
class PingPongCoacher extends Coacher implements English {
	//构造方法
	public PingPongCoacher(){}
	public PingPongCoacher(String name){
		super(name);
	}
	//重写所有的抽象方法
	public void teach(){
		System.out.println("教如何扣球");
	}
	public void eat(){
		System.out.println("吃鲍鱼");
	}
	public void studyEnglish(){
		System.out.println("学英语");
	}
}

//(具体类)足球教练 extends 教练
class FootballCoacher extends Coacher {
	//构造方法
	public FootballCoacher(){}
	public FootballCoacher(String name){
		super(name);
	}
	//重写所有的抽象方法
	public void teach(){
		System.out.println("教如何踢假球");
	}
	public void eat(){
		System.out.println("吃鸡骨头");
	}
}

class InterfaceTest3 {
	public static void main(String[] args) {
		//乒乓球运动员
		PingPongPlayer p = new PingPongPlayer("小明");
		System.out.println(p.getName() );
		p.study();
		p.eat();
		p.studyEnglish();
	}
```

```
}
```

二、分析题

1、写出下面程序的执行结果和执行过程。

```
class X{
    Y b = new Y();//先完成对象成员的创建，再执行构造方法  //5
    X(){
            System.out.print("X");    //8
        }
    }
class Y{
    Y(){        //6
            System.out.print("Y");    //7
        }
    }
public class Z extends X{
    Y y = new Y();  //10
    Z(){        //3
        //super();//此处隐藏了父类的无参构造函数 4
        System.out.print("Z");    //9
    }
    public static void main(String[] args){//1
        new Z();   //2
    }
}
```

答案：YXYZ

创建对象时，先执行父类构造方法，再执行子类构造方法；

创建对象调用构造方法时，先完成对象成员的创建，再执行构造方法

2、写出下代码的运行结果和执行过程。

```
class Demo {
    static int i=1, j=2;
    static {
        display(i);//1
    }    //在同一个类中静态代码块优先执行
    public static void main(String[] args) {
        display(j);//3
    }
    public static void display(int n) {
        System.out.println(n);          //2、4
    }
}
```

答案：
1
2
先执行静态代码块，再执行静态函数

3、写出下代码的运行结果和执行过程。

```
public Demo {
    static int total = 10;
    public static void main(String[] args){
        new Demo();//匿名对象只使用一次
        new Demo();
    }
    public Demo(){
        int temp = this.total;
        if (temp++ > 5) {
            System.out.println(temp);
        }
    }
}
```

答案：11　　//匿名对象只是用一次

本文是面向对象的学习笔记之二的相关测试代码，如有错误，还请大家指出，谢谢!!!