南京工业职业技术大学试卷

面向对象高级程序设计考试

2020/2021 学年第二学期

试卷类别：B卷 考试方式：闭 卷 适用班级：软件 2014/2024/2034/2010/2020

命 题 人：_____ 审 核 人：_____ 考试性质：期末

班 级：_____ 学 号：_____ 姓 名：_____

题号	一	二	三					总分
得分								

一、单项选择题（共 10 题，每题 2 分，共 20 分）

1．下面关于类及其修饰符的描述，不正确的是（ B ）。
A.abstract 类只能用来派生子类，不能用来创建 abstract 类的对象
B.final 类不但可以用来派生子类，也可以用来创建 abstract 类的对象
C.abstract 不能与 final 同时修饰一个类
D. abstract 方法必须在 abstract 类中声明，但 abstract 类定义中可以没有 abstract 方法

2．下列选项中关于 Java 中 super 关键字的说法正确的是（ A ）。
A.super 关键字是在子类对象内部指代其父类对象的引用
B.super 关键字不仅可以指代子类的直接父类，还可以指代父类的父类
C.子类通过 super 关键字只能调用父类的方法，而不能调用父类的属性
D.子类通过 super 关键字只能调用父类的属性，而不能调用父类的方法

3．在 Java 中，下面对于构造方法的描述正确的是（ D ）。
A.类必须显示定义构造方法
B.构造方法的返回类型是 void
C.构造方法和类有相同的名称，并且不能带任何参数
D.一个类可以定义多个构造方法

4．实现下列（ D ）接口，可以启用比较功能。
A.Runnable 接口 B.Iterator 接口 C.Serializable 接口 D.Comparator 接口

5. 已知父类 Person，子类 Man。判断类 Person 的对象 person1 是否满足类 Man 的实例特征，正确的语句为（ A ）。

Person person1 = new Man();
Man man1 = new Man();

A. if (person1 instanceof Man)　　B. if (man1 instanceof Person)
C. if (Person instanceof man1)　　D. if (Man instanceof person1)

6. 下列选项中（ C ）不属于 I/O 流？
A.FileWriter　　B. FileReader　　C.Properties　　D.PrintStream

7. 下列选项中，关于 Java 的抽象类和抽象方法说法不正确的是（ B ）。
A. 抽象类和抽象方法都通过 abstract 关键字来修饰
B. 抽象类中必须包含抽象方法
C. 抽象方法只有方法声明，没有方法实现
D. 子类如果不重写父类所有的抽象方法，则必须设置为抽象类

8.下面的方法重载，正确的是（ C ）。
A.int fun(int a,float b){}
float fun(int a,float b){}
B.float fun(int a,float b){}
float fun(int x,float y){}
C.float fun(float a){}
float fun(float a,float b){}
D.float fun1(int a,float b){}
float fun2(int a,float b){}

9.关于异常处理机制，正确的说法是（ B ）。
A.catch 部分捕捉到异常情况时，才会执行 finally 部分
B.当 try 区块的程序发生异常时，才会执行 catch 区块的程序
C.不论程序是否发生错误及捕捉到异常情况，都不会执行 finally 部分
D. 以上都是

10. 线程通过（ C ）方法可以休眠一段时间，然后恢复运行。
A.run　　B.setPrority　　C.sleep　　D.yield

二、读程序题 （共 4 题，每题 5 分，共 20 分）

1. 以下程序的输出结果为___输出结果：2, 4, 6, 8, 10, 10, 8, 6, 4, 2_____。

```
public class Test1 {
public static void main(String[] args) {
    ArrayList<Integer>  list=new ArrayList<>();
      list.add(2);
      list.add(4);
      list.add(6);
      list.add(8);
      list.add(10);
      mirror(list);
      System.out.print(list);
```

```
    }
     public static    void mirror(ArrayList<Integer>    list) {
          for(int i=list.size()-1;i>=0;i--) {
               list.add(list.get(i));
           }
     }
}
```

2. 以下程序的输出结果为___输出结果：new a shape new a circle_____。

```
    class Shape{
          public Shape(){
               System.out.print("new a shape");
          }
     }
    class Circle extends Shape{
          public Circle(){
               System.out.print("new a circle");
     }
}
   public class Test2 {
          public static void main(String[] args) {
               Shape s=new Circle();
          }
}
```

3. 以下程序的输出结果为__输出结果：IndexOutOfBoundsException_____。

```
public class Test3 {
     public static void main(String[] args) {
               try {
                    int[ ] list=new int[10] ;
                    System. out.println ("list[10] is "+list [10]) ;
               }
               catch( ArithmeticException ex ) {
```

```
                System.out.println("ArithmeticException");
            }
            catch(IndexOutOfBoundsException ex ){
                System.out.println ("IndexOutOfBoundsException") ;
            }
            catch(Exception ex ) {
                System.out.println ("Exception") ;
            }
        }
    }
```

4. 以下程序的输出结果为_____。

```
class Animal {
    public void show() {
        System.out.println("我是动物");
    }
}
class Tiger extends Animal {
    public void show() {
        System.out.println("我是老虎");
    }
}
class Dog extends Animal {
    public void show() {
        System.out.println("我是哈士奇");
    }
}
public class Test4 {
    public static void main(String[] args) {
        Animal one = new Animal();
        one.show();
        Animal two = new Tiger();
        two.show();
        Animal three = new Dog();
        three.show();
    }
}
```

输出结果：

我是动物

我是老虎

我是哈士奇

三、编程题 （共 4 题，每题 15 分，共 60 分）

1. 编写程序 SetList.Java，创建一个 ArrayList, 然后向这个列表中添加一个 Date 对象、一个字符串和一个 Circle 对象，然后循环调用对象的 toString()方法，以显示列表中的所有元素。Circle 类定义如下：

```
class Circle {
        private    int radius;
        public Circle(int radius) {
                this.radius = radius;
        }
        public    double Area(){
            double s;
            s=Math.PI*this.radius*this.radius;
            return s;
        }
        public String toString(){
            return    “此圆的半径是”+this.radius+“此圆的面积是”+this.Area();
        }
    }
```

```
//Handset (2 分)
   public abstract class Handset {
        private String brand;
        private String type;
        public Handset(String brand,String type){
                this.brand=brand;
                this.type=type;
        }
        public abstract void sendInfo();
        public abstract void call();
        public void info(){
              System.out.println("这是一款型号为"+type+"的"+brand+"手机:");
        }
    }
     //Network 接口（2 分）
      public interface Network {
```

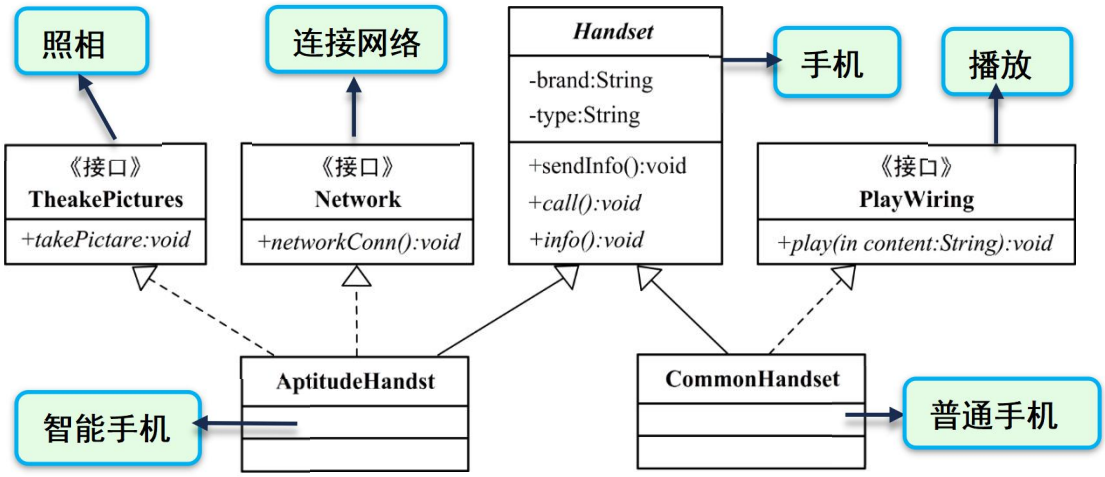
```
        void networkConn();
    }
    //PlayWriting 接口（2 分）
     public interface PlayWiring {
        void play(String content);
    }
    //TheakePicture 接口（2 分）
    public interface TheakePictures {
        void takePicture();
    }
    // AptitudeHandset 类（4 分）
public class AptitudeHandset extends Handset implements Network, PlayWiring,TheakePictures
{
    public AptitudeHandset(String brand,String type){
         super(brand,type);
    }
    @Override
    public void sendInfo() {
          System.out.println("发送带图片与文字的信息……);
    }
    @Override
    public void takePicture() {
          System.out.println("咔嚓.....拍照成功");
    }
   @Override
   public void play(String content) {
          System.out.println("开始播放视频《"+content+"》.....");
   }
   @Override
   public void networkConn() {
```

```
            System.out.println("已经启动移动网络.....");
    }
    @Override
    public void call() {
        System.out.println("开始视频通话.....");
    }
}
// CommonHandset 类（3 分）
public class CommonHandset extends Handset implements PlayWiring {
    public CommonHandset(String brand,String type){
        super(brand,type);
    }
    @Override
    public void play(String content) {
        System.out.println("开始播放音乐《"+content+"》.....");
    }
    @Override
    public void sendInfo() {
        System.out.println("发送文字信息.....");
    }
    @Override
    public void call() {
        System.out.println("开始语音通话.....");
    }
}
```

2. 编写类及接口，参照以下类的结构图以及给出的测试类，让普通手机播放音频、发信息和通电话，让智能手机上网、播放视频、照相、发信息和通电话。其中 info 方法实现输出手机的型号和品牌。其他方法实现输出其相应的功能即可。例如：

```
@Override
public void play(String content) {
```

```
        System.out.println("开始播放视频《"+content+"》.....");
    }
```



测试类：

```
public class HandsetTest {
    public static void main(String[] args) {
        CommonHandset coHandset=new CommonHandset("索尼爱立信","G502c");
        coHandset.info();
        coHandset.play("最炫民族风");
        coHandset.sendInfo();
        coHandset.call();
        System.out.println();
        AptitudeHandset aHandset=new AptitudeHandset("华为","Honor 20");
        aHandset.info();
        aHandset.networkConn();
        aHandset.play("觉醒年代");
        aHandset.takePicture();
        aHandset.sendInfo();
        aHandset.call();
    }
}

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

```
                ArrayList<Object>    lists=new ArrayList<Object>();      （2 分）
                Circle c1=new Circle(2);                                 （2 分）
                Date date=new Date();                                    （2 分）
                lists.add(c1);                                           （1 分）
                lists.add(date);                                         （1 分）
                lists.add("yangjingli");                                 （1 分）
                for(Object l:lists) {                                    （6 分）
                    System.out.println(l);
                }
        }
}
```

3. 写两个线程，一个线程(TPNumber)打印 1~100，另一个线程(TPChar)打印 A~Z，在测试类 Demo 中启动这两个线程。

```
class TPNumber extends Thread{                                           （5 分）
    public void run() {
        for(int i=1;i<100;i++) {
            System.out.println(i);
        }
    }
}
class TPChar    extends Thread{                                          （5 分）
    public void run() {
        for(char ch='A';ch<='Z';ch++) {
            System.out.println(ch);
        }
    }
}
public class Demo {                                                      （5 分）
        public static void main(String[] args) {
            TPNumber    tp1=new TPNumber();
            TPChar tp2=new TPChar();
            tp1.start();
```

```
            tp2.start();
        }
}
```

4. 假设现有本机数据库 PetM，其中有表 pet(id, masterid, name, health,love)，其中 id、masterid、name 均为字符串类型，health、love 均为整型，请通过 jdbc 对该表进行操作，将名字 name 字段为“王牌”的宠物从宠物表 pet 中删除。

```
public class DD {
    static String url="jdbc:mysql://localhost:3306/petM";                    （3 分）
    static String user="root";
    static String password="root";
    static Connection connection=null;
    static PreparedStatement    pst=null;
    public static void main(String[] args) {
          try {
              Class.forName("com.mysql.jdbc.Driver");                        （2 分）
          } catch (ClassNotFoundException e) {
            e.printStackTrace();
          }
        try {
              connection=DriverManager.getConnection(url, user, password);   （2 分）
            } catch (SQLException e) {
                  e.printStackTrace();
            }

      try {
              pst=connection.prepareStatement(sqlString);                    （3 分）
              pst.executeUpdate();                                           （3 分）
          } catch (SQLException e) {
                e.printStackTrace();
}
```

南京工业职业技术大学试卷

面向对象程序设计 I(2)考试

2022/2023 学 年 第 一 学 期

试卷类别： A 卷 考试方式： 闭 卷 适用班级： 软件 2231/32/41

命 题 人：_____审 核 人：_____考试性质： 期末

班 级：_____学 号：_____姓 名：_____

题号	-	二	三	四	五			总分
得分								

一、单项选择题（共 10 题，每题 2 分，共 20 分）

1. 以下关于变量的说法错误的是？ （ C ）

A. 变量名必须是一个有效的标识符。

B. 变量在定义时可以没有初始值。

C. 变量一旦被定义，在程序中的任何位置都可以被访问。

D. 在程序中，可以将一个 byte 类型的值赋给一个 int 类型的变量，不需要特殊声明。

2. 下面关于子类调用父类构造方法的描述正确的是（ C ）。

A. 子类定义了自己的构造方法，就不会调用父类的构造方法

B. 子类必须通过 super 关键字调用父类有参的构造方法

C. 如果子类的构造方法没有通过 super 调用父类的构造方法，那么子类会先调用父类中无参构造方法，之后再调用子类自己的构造方法

D. 创建子类对象时，先调用子类自己的构造方法，然后再调用父类的构造方法

3. 编译并运行下面的程序，结果是（ B ）。

```
public class A {
    public static void main(String args[]) {
        B b = new B();
        b.test();
    }
    void test() {
        System.out.print("A");
```

```
        }
    }
    class B extends A {
        void test() {
            super.test();
            System.out.println("B");
        }
    }
```

A．产生编译错误　　B．代码可以编译运行，并输出结果：AB

C．代码可以编译运行，但没有输出　D．编译没有错误，但会运行时会产生异常

4. 要想集合中保存的元素没有重复并且按照一定的顺序排列，可以使用以下哪个集合？（ D ）

A．LinkedList　B．ArrayList　C．hashSet　D．TreeSet

5. 使用那个关键字可以在程序中手工抛出异常（ A ）。

A．throw　B．throws　C．assert　D．class

6. 线程调用 sleep()方法后，该线程将进入以下哪种状态？（ A ）

A．阻塞状态　B．运行状态

C．就绪状态　D．死亡状态

7. 下列异常属于可检测的是（ C ）。

A．ArithmeticException　B．IndexOutOfBoundsException

C．SQLException　D．NullPointerException

8. 假设类 X 是类 Y 的父类，下列声明对象 x 的语句中不正确的是（ D ）。

A．X x = new X() ;　B．X x = new Y() ;

C．Y x = new Y() ;　D．Y x = new X() ;

9. 下面关于 String 类的特点描述正确的一项是？（ B ）

A．String 类在需要时可以定义子类

B．String 类的对象内容一旦声明则不可改变

C．String 类可以直接利用“==”进行字符串内容的比较

D．String 类对象实例化后都会自动存入字符串对象池

10. PreparedStatement 接口执行 executeQuery()方法后返回的数据类型是（ A ）。

A．ResultSet 接口对象　B．Statement 接口实例

C．Connection 接口实例　D．DatabaseMetaData 类的对象

二、 填空题（共 10 空，每空 1 分， 共 10 分）

1. 从 JDK 1.8 之中，接口内可以定义有三类方法：抽象方法、 __default 定义的普通方法__ 、 static 定义的 静态方法

2. Object 类是所有类的父类，该类中判断两个对象是否相等的方法是___equals()或 boolean equals(Object o)___。

3. 接口中的方法被默认的访问权限是____public___。

4. Java 语言中， 表示一个类 A 继承自父类 B，并实现接口 C 的语句是_____class A extends B implements C_____。

5. ArithmeticException 类表示___算术___ 异常，RunTimeException 表示__运行时___ 异常。

6. 数组声明后，必须使用___new__运算符分配内存空间。

7. 将父类对象转换为子类对象时，必须使用____强制类型转换____。

8. 使当前线程休眠 4 秒， 其实现语句为____Thread.sleep(4000)__。

9. 在JavaJDBC编程中， ___PreparedStatement__ 接口的对象可以代表一个预编译的 SQL语句，它是 Statement接口的子接口。

三、 程序填空题 （共 4 题， 每题 5 分，共 20 分）

1. 以下程序的输出结果为___num=12___。

```
public class Demo1 {
        public static void main(String args[]) {
            char c = 'A' ;
            int num = 10 ;
            switch(c) {
                case 'B' :
                    num ++ ;
                case 'A' :
                    num ++ ;
                case 'Y' :
                    num ++ ;
                    break ;
                default :
                    num -- ;
            }
            System.out.println("num="+num) ;
        }
    }
```

2. 以下程序的输出结果为_____。

```
3. public class Demo2 {
        public static void main(String[] args) {
            int i;
            int[] a = {1, 2, 3, 4};
            for(i=0;i<5;i++){
             try {
```

```
捕获到算术异常！
finally i=0
a[1]/1=2

finally i=1
a[2]/2=1
finally i=2
a[3]/3=1
finally i=3
捕获到了数组下标越界异常！
finally i=4

继续。。。
```

```
            System.out.println("a[" + i + "]/" + i + "=" + (a[i] / i));
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("捕获到了数组下标越界异常！  ");

        }catch (ArithmeticException e){
            System.out.println("捕获到算术异常！  ");
        } finally{
            System.out.println("finally i=" + i);
         }
       }
        System.out.println("继续。。。  ");
    }
}
```

3. 以下程序的输出结果为_____。

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class Demo3 {
    public static void main(String[] args) {
        List<String> list=new ArrayList<>();
        list.add("Tom");
        list.add("Jerry");
        list.add(1, "Alpha");
        list.add("Dota");
        list.add("Jack");
        list.add(1, "Tomy");
        Iterator<String> iterator=list.iterator();
        while(iterator.hasNext()){
            if(iterator.next().startsWith("J")){
                iterator.remove();
            }
        }
        list.forEach(System.out::println);
    }
}
```

```
Tom
Tomy
Alpha
Dota
```

4. 根据提示填写程序中的空白_____。

```
interface Achievement {
```

```
（1） double average();
（2） extends Person implements Achievement
（3） super(name, age);super(name, age);
（4） s.introduce();
（5） System.out.println(s.average());
```

```
    （1）//   声明一个返回double 类型的方法average()
    }

class Person {
    private String name;
    private int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public void introduce(){
        System.out.println("您好！我是"+name+",今年"+age+"岁");
    }
}
class Student （2）//继承Person 并实现Achievement 接口 {
    private int java;
    private int english;

    public Student(String name, int age) {
        （3）//调用父类的构造方法Person(String name,int age)

    }
    public void setScore(int java,int english){
        this.java=java;
        this.english=english;
    }

    @Override
    public double average() {
        return (java+english)/2.0;
    }
}
public class Demo4 {
    public static void main(String[] args) {
        Student s = new Student("张三", 19);
        （4）//调用s 的introduce()方法
        s.setScore(90,85);
        （5）//显示s 的平均分
```

```
    }
}
```

四、 简答题（共 3 题， 每题 5 分，共 15 分）

1. 什么是多态？实现多态的方法有哪些？

多态是面向对象的三大特征之一。 同一个行为具有不同表现形式或形态的能力

多态实现的方式：接口实现、继承父类进行方法的重写、在同一个类中进行方法重载

2. 写出类集 ArrayList 的至少 3 种遍历方式，并举例说明。

答出三种即可

（1） 普通的 for 语句

（2） 增强 for 循环

（3） 迭代器方式 Iterator、双向迭代器 ListIterator

（4） forEach() Iterable 接口中的默认方法

（5） Stream API 中的 forEach()

3. 写出读取一个关系型数据库的步骤。

（1）加载数据驱动

（2）建立与数据库的连接对象

（3）建立一个语句对象

（4）执行 sql 语句

（5）处理结果积极

（6）关闭所有 JDBC 对象

五、 编程题 （共 3 题， 35 分）

1. 创建一个Student类， 具有封装的属性name:String，age：int，要求：

（1）创建属性的setter和getter方法、无参构造方法以及全参构造方法、编写toString()方法。 （2分）

（2）利用全参构造方法创建5个Student对象，要求包含有同名的(英文)、同龄的，并将其存储到类集ArraysList类型的list之中。(3分)

（3）对list进行过滤，将年龄大于18的数据进行输出。(4分)

（4）利用ArraysList类集的sort()方法对该list进行排序， 排序规则为： 按年龄从小到大进行排列， 同龄的按name 的英文字典顺序进行。并将排序后的list进行输出(6分)

参考代码：

```
public class Student {
    private String name;
    private int age;
```

```
    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;
    }

    @Override
    public String toString() {
        return "Student{" +
                "name='" + name + '\'' +
                ", age=" + age +
                '}';
    }

    public Student() {
    }

     public Student(String name, int age) {
         this.name = name;
         this.age = age;
     }
}
public class StudentTest {
     public static void main(String[] args) {
         Student s1 = new Student("Tom", 20);
         Student s2 = new Student("Jerry", 18);
         Student s3 = new Student("Tom", 22);
         Student s4 = new Student("Jack", 18);
         Student s5 = new Student("Lily", 19);
```

```
        ArrayList<Student> list=new ArrayList<>();
        list.add(s1);
        list.add(s2);
        list.add(s3);
        list.add(s4);
        list.add(s5);
        list.stream().filter(s->s.getAge()>18).forEach(System.out::println);
        System.out.println("sorted:");
        list.sort((o1,o2)->{
            int m=o1.getAge()-o2.getAge();
            int n=(m==0)?o1.getName().compareTo(o2.getName()):m;
            return n;
        });
        list.forEach(System.out::println);
    }
}
```

2. 按以下描述进行程序设计：

（1） 设计一个抽象类Shape，包含有两个抽象方法： 计算面积getArea()，返回值类型为double 。计算周长getPerimeter()，返回值类型为double（2分）

（2） 定义一个接口IPoint，包含有一个方法：计算体积getVolume()，返回值类型为double。(2分)

（3） 定义一个圆类Circle，继承抽象类Shape，并具有封装的属性radius：int，实现getArea()方法计算Circle的面积。实现getPerimeter()方法计算Circle的周长(3分)

（4） 定义一个圆柱类Cylinder，继承类Circle，实现接口Ipoint，增加了有封装的属性height，覆写getArea()求圆柱的表面积， getVolume()返回圆柱的体积（提示：圆柱的表面积公式为 $2\pi r*h+2\pi r^2$，圆柱的体积公式为：πr^2h，其中r代表半径，h代表高）。(3分)

（5） 编写测试类，利用构造方法创建一个带圆半径的圆对象c1和一个带圆柱体底半径和圆柱高的圆柱对象c2，输出其c1的面积和c2的表面积及体积。(2分)

参考代码：

```
abstract class Shape {
    abstract    double getArea();
    abstract    double getPerimeter();
}
interface IPoint {
    double getVolume();
}
```

```
class Circle extends Shape{
	private int radius;
	public Circle(int radius) {
		this.radius = radius;
	}
	@Override
	double getArea() {
		return Math.PI*radius*radius;
	}
	@Override
	double getPerimeter() {
		return 2*Math.PI*radius;
	}
}
class Cylinder extends Circle implements IPoint{
	private int height;
	public Cylinder(int radius,int height) {
		super(radius);
		this.height=height;
	}
	@Override
	double getArea() {

			return super.getArea()*2+super.getPerimeter()*height; }
		@Override
		public double getVolume() {
			return super.getArea()*height;
		}
	}
	public class Demo02 {
		public static void main(String[] args) {
			Circle circle=new Circle(1);
			Cylinder cylinder = new Cylinder(1, 3);
			System.out.println(circle.getArea());
			System.out.println(cylinder.getArea());
			System.out.println(cylinder.getVolume());
		}
	}
```

4. 编写程序实现以下功能，一个线程实现输出 1-26，另一个线程实现输出 a-z，实现两个 线程的轮流输出。（8 分）

参考代码：

```
public class ThreadDemo02Runnable {
        public static void main(String[] args) {
                new Thread(new Runnable() {
                        @Override
                        public void run() {
                                for (int i = 0; i < 26; i++) {
                                        System.out.print(i+"    ");
                                        Thread.yield();
                                }
                        }
                }).start();
                new Thread(()->{
                        for(int i=97;i<123;i++){
                        System.out.print((char)i+"    ");
                        Thread.yield();
                }}).start();

        }
}
```

南京工业职业技术大学试卷

面向对象程序设计 I(2)考试

2022/2023 学 年 第 1 学期

试卷类别： B卷 考试方式： 闭 卷 适用班级： 软件 2231、软件 2232 、软件 2241

命 题 人：_____ 审 核 人：_____ 考试性质： 期末

班 级：_____ 学 号：_____ 姓 名：_____

题号	-	二	三	四	五	六	七	总分
得分								

一、 单项选择题(共 10 题， 每题 2 分，共 20 分)

1. 下列叙述中，正确的是（ C ）。
   A. 子类继承父类的所有属性和方法
   B. 子类可以继承父类私有的属性和方法
   C. 子类可以继承父类公有的属性和方法
   D. 创建子类对象时， 父类的构造方法都要被执行
2. 关于 Java 中的多态，以下说法中不正确的是（ B ）。
   A. 多态不仅可以减少代码量，还可以提高代码的可扩展性和可维护性
   B. 把子类转换为父类，称为向下转型，自动进行类型转换
   C. 多态是指同一个实现接口，使用不同的实例而执行不同的操作
   D. 继承是多态的基础，没有继承就没有多态
3. 给定如下Java程序，Test类中的三个输出语句输出结果依次是（ C ）。

```
class Person {
    private String name="person";
    public void shout(){
        System.out.println(name);
    }
}
class Student extends Person{
    private String name="student";
```

```
    private String school="school";
}
public class Test {
    public static void main(String[] args) {
        Person p=new Student();
        System.out.println(p instanceof Student);
        System.out.println(p instanceof Person);
        System.out.println(p instanceof Object);
    }
}
```

A. true false true　　B. false true false

C. true true true　　D. 编译错误

4. 下面关于抽象类和接口的论述，正确的是（ C ）。

A．接口中可以包含普通方法

B．抽象类中只能包含抽象方法， 不能包含普通方法

C．接口中只能定义静态常量属性，不能定义普通属性

D．接口中可以包含构造器

5. 如下所示的 Sample 类中，共有（ B ）个构造方法。

```
public class Sample{
    private int x;
    public Sample(){ x=1;}
    public void Sample(double f){this.x=(int) f;}
    public Sample(String s){this.x=Integer.parseInt(s);}
}
```

A. 1　　B. 2　　C.3　　D. 4

6. 下列选项中关于 Java 中封装的说法错误的是（ D ）。

A．封装就是将属性私有化，提供共有的方法访问私有属性

B．属性的访问方法包括setter方法和getter方法

C．setter方法用于赋值， getter方法用于取值

D．包含属性的类都必须封装属性， 否则无法通过编译

7. 对于 catch 子句的排列，下列（ A ）种是正确的？

A．子类在先，父类在后

B．父类在先，子类在后

C．有继承关系的异常不能在同一个try程序段内

D．先有子类，其他如何排列都无关

8. 在 Java 中开发 JDBC 应用程序时， 使用 DriverManager 类的 getConnection()方法建立与数据源的连接语句为：

Connection con = DriverManager.getConnection("jdbc:mysql:news");URL 连接中的“news”

表示的是（ C ）。

A. 数据库中表的名称　　B. 数据库服务器的机器名

C. 数据源的名称　　D. 用户名

9. JDBC 执行 SQL 查询后返回的结果类型是（ D ）。

A. Statement　　B. DriverManager

C. PreparedStatement　　D. ResultSet

10.当前线程等待另一个线程执行完，再继续执行的方法是（ B ）。

A. sleep　　B. join　　C. interrupt　　D. notify

二、 填空题（每空 1 分，共 10 分）

1. 面向接口编程是___多态__ 的一种体现，面向接口编程是使用接口来约束类的行为。

2. 访问控制修饰符说明类或类的成员的可访问范围，用 ____public____ 修饰的类或成员拥有 公共作用域，表明此类或类的成员可以被任何 Java 中的类所访问。

3. 子类必须通过 ____super___ 关键字调用父类有参数的构造方法。

4. Java 语言中，表示一个类 A 继承自父类B，并实现接口 C 的语句是____class A extends B implements ___。

5. 已经定义了 Student 类，创建一个长度为 10 的 Student 类型的数组 students 的语句是 ___Student[] students = new Student[10];____ 。创建一个泛型为 Student 且元素不可重复集合 students 的语句是__Set<Student> students = new HashSet<>();____。

6. _____abstract____ 方法是一种仅有方法头，没有具体方法体和操作实现的方法， 该方法必须在抽象类中定义。 ____final__ 方法是不能被当前类的子类重新定义的方法。

7. 当线程调用 start()方法后， 其所处状态为_____就绪状态___。

8. 在 Java JDBC 编程中， 可以使用PreparedStatement 接口为特定的 SQL 命令指定多个 参数， 此时需要在创建 SQL 语句时为每个参数各用一个____?___ 符号为占位符。

三、 读程序题 （共 4 题，每题 5 分，共 20 分）

1. 以下程序的输出结果为_____。

```
B's constructor is invoked
A's constructor is invoked
```

```
class A extends B {
    public A(int t){
        System.out.println("A's constructor is invoked");
    }
}
class B {
    public B(){
```

```
            System.out.println("B's constructor is invoked" ) ;
        }
    }
    public    class    test1 {
        public static void main(String[] args) {

                A   a=new A(3);
        }
    }
```

2. 以下程序的输出结果为_____。

```
RuntimeException in method()
After the method call
```

```
public class test2 {
    public static void main(String[] args) {
        try {
                method() ;
                System.out.println ( "After the method call" ) ;
          } catch(RuntimeException ex ) {
                System.out.println ( "RuntimeException in main " ) ;
          } catch(Exception ex ){
                System.out.println("Exception in main");
        }
    }
    public static void method ( )throws Exception    {
        try {
                String s="abc";
                System.out.println (s.charAt(3)) ;
          }catch( RuntimeException ex ) {
                System.out.println("RuntimeException in method()");
          }catch( Exception ex ) {
                System.out.println(" Exception in method()" ) ;
        }
    }
}
```

3. 以下程序的输出结果为_____。

```
size= 3
18
38
8
```

```
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;
 public class test3{
        public static void main(String[] args){
                Set    set=new HashSet<>();
                set.add(new Integer(8));
```

```
                set.add(new Integer(18));
                set.add(new Integer(38));
                set.add(new Integer(8));
                set.add(new Integer(18));

                System.out.println("size= "+set.size());
                Iterator it=set.iterator();
                while(it.hasNext()) {
                    System.out.println(it.next()+ "    ");
                }
            }
        }
```

4. 以下程序的输出结果为____6____。

```
import java.util.ArrayList;
public class test4 {
    public static void main(String[] args) {
            ArrayList<Integer> list = new ArrayList<>();
            for (int k = 1; k <= 10; k++) {
                list.add(new Integer(k));
            }
            list.remove(3);
            list.remove(3);
            Integer m = list.get(3);
            System.out.println(m.intValue());
        }
}
```

四、 简答题（共 3 题，每题 5 分， 共 15 分）

1. 简述 this 和 super 关键字的用处。

this 关键字表示当前对象，主要用于：

1）调用成员变量
2）调用构造方法
3）调用普通方法
4）返回当前对象

super 关键字表示父类对象，主要用于：

1）调用父类的构造方法
2）调用父类的成员属性
3）调用父类的非私有成员方法

2. 什么是 RuntimeException？列举至少 4 个 RuntimeException 的子类。

RuntimeException 是不检查异常，程序中可以选择捕获处理， 也可以不处理。这些异常一般是由程序逻辑错误引起的，程序应该从逻辑角度尽可能避免这类异常的发。当出现 RuntimeException 的时候，我们可以不处理。当出现这样的异常时， 总是由虚拟机接管。RuntimeException 是java 中所有运行时异常的父类， 实际运行时出现的都是它的子类。

ArithmeticException 类、ArrayIndexOutOfBoundsException 类、ClassCastException 类、IllegalArgumentException 类、IndexOutOfBoundsException 类等。

3. 简述 JDBC 编程的步骤。

（1） 加载驱动程序

（2） 建立连接

（3） 创建语句

（4） 执行语句

（5） 处理 ResultSet

（6） 关闭连接

五、 编程题 （共 3 题， 35 分）

1. 已知手机类Phone ，完成下面各题。

```
public class Phone {
    private String brand;
    private int price;
    private String color;
    public Phone(String brand, int price,String color) {
            this.brand = brand;
            this.price = price;
            this.color = color;
    }
    public String getBrand() {
```

```
            return brand;
        }
        public void setBrand(String brand) {
            this.brand = brand;
        }
        public   String   toString() {
            return "phone brand:"+this.brand+"price:"+this.price+"color:"+this.color;
        }
    }
```

（1） 创建5个手机对象，存储到HashMap集合中，手机品牌brand作为键， 手机对象作为值。（5分）

（2） 遍历输出集合中所有手机的信息。（5分）

（3） 从键盘输入一个手机品牌，检测集合中是否存在此品牌的手机，存在的话， 打印该手机信息。（5分）

```
class Phone {
    private String brand;
    private int price;
    private String   color;
    public Phone(String brand, int price, String   color) {
        this.brand = brand;
        this.price = price;
        this.color = color;
    }
    public String getBrand() {
        return brand;
    }
    public void setBrand(String brand) {
        this.brand = brand;
    }
    public   String toString() {
        return "phone brand:"+this.brand+"price:"+this.price+"color:"+this.color; }
}
public class MapPhone{
    public static void main(String[] args) {
        //5分   创建5个手机对象，存储到HashMap集合中
        Phone phone1=new Phone("苹果", 6000, "black");
        Phone phone2=new Phone("小米", 2000, "white");
        Phone phone3=new Phone("华为", 3000, "red");
```

```
        Phone phone4=new Phone("vivo", 3500, "black");
        Phone phone5=new Phone("三星", 1900, "red");
        HashMap<String, Phone> phones=new HashMap<>();
        phones.put(phone1.getBrand(), phone1);
        phones.put(phone2.getBrand(), phone2);

        phones.put(phone3.getBrand(), phone3);
        phones.put(phone4.getBrand(), phone4);
        phones.put(phone5.getBrand(), phone5);
        //5分  遍历输出集合中所有手机的信息
        for(String key:phones.keySet()) {
            System.out.println(phones.get(key));
        }
        //5分
        Scanner input=new Scanner(System.in);
        System.out.println("请输入手机品牌：  ");
        String brand=input.next();
        if(phones.containsKey(brand)) {
            System.out.println(phones.get(brand));
        }
    }
}
```

2. 设计接口IFlyable（会飞的），包含一个方法fly()，该方法返回值为String。（2分）设计抽 象类Bird，包含一个抽象方法bark()，表示鸟类的“叫”，返回一个字符串（2分）。设计麻 雀类Sparrow类，继承自Bird类， bark()方法返回字符串“a sparrow is barking”；并实现接口 IFlyable，fly()方法返回字符串“a sparrow is flying”。（3分）设计鹦鹉类Parrot类， 继承自Bird 类， bark()方法返回字符串“a parrot is barking”，并实现接口IFlyable ，fly()方法返回字符串 “a parrot is flying”。（3分）设计测试类，定义Sparrow，Parrot对象， 并调用各自方法。（2分）

```
//2分
    interface IFlyable{
        public String fly();
    }
     //2分
    abstract class Bird{
        public abstract String bark();
     }
```

```
  //3分
  class Sparrow    extends Bird implements IFlyable{
     @Override
        public String fly() {
             return "a sparrow is flying";
     }
     @Override
     public String bark() {
           return "a sparrow is barking";
     }
 }

  //3分
  class Parrot extends Bird implements IFlyable{
     @Override
     public String fly() {
           return "a parrot is flying";
     }
     @Override
     public String bark() {
           // TODO Auto-generated method stub
           return "a parrot is barking";
     }
}
 //2分
 public class fly {
     public static void main(String[] args) {
           Parrot   parrot=new Parrot();
           parrot.fly();
           parrot.bark();
           Sparrow sparrow=new Sparrow();
           sparrow.fly();
           sparrow.bark();
     }
}
```

3. 写两个线程轮流打印数字， 从 1 到 100 。(8 分)

```
 public class TakeTurnsPrint {
     public static class TakeTurns{
           private static boolean flag=true;
```

```
        private static int count=0;
        // 2分
        public synchronized void print1() {
            for(int i=1;i<=50;i++) {
                while(!flag) {

                try {
                    System.out.println("print1:wait before");
                    wait();
                    System.out.println("print1:wait after");
                } catch (InterruptedException e) {

                    e.printStackTrace();
                }
            }
            System.out.println("print1:"+ ++count);
            flag=!flag;
            notifyAll();
        }
    }
// 2分
public   synchronized   void print2() {
    for(int i=1;i<=50;i++) {
        while(flag) {
            try {
                System.out.println("print2:wait before");
                wait();
                System.out.println("print2:wait after");
            } catch (InterruptedException e) {

                e.printStackTrace();
            }
        }
        System.out.println("print2:"+ ++count);
        flag=!flag;
        notifyAll();
    }
    }
}
// 4分
```

```
public static void main(String[] args) {
        TakeTurns takeTurns=new TakeTurns();

            new Thread(new Runnable() {
                    @Override
                public void run() {

                    takeTurns.print1();
                }
            }).start();
            new Thread(new Runnable() {

                @Override
                public void run() {
                    takeTurns.print2();
                }
            }).start();
    }
}
```