继承作业

一、 选择题

1. 以下关于继承条件下构造方法执行过程的代码的执行结果是（ A ）。（选择一项）

```
class Person {
    public Person() {
        System.out.println("execute Person()");
    }
}
class Student extends Person {
    public Student() {
        System.out.println("execute Student() ");
    }
}
class PostGraduate extends Student {
    public PostGraduate() {
        System.out.println("execute PostGraduate()");
    }
}
public class TestInherit {
    public static void main(String[] args) {
        new PostGraduate();
    }
}
```

A. execute Person()
execute Student()
execute PostGraduate()

B. execute PostGraduate()

C. execute PostGraduate()
execute Student()
execute Person()

D. 没有结果输出

2. 以下关于 this 和 super 关键字的说法错误的是（ BD）。（选择二项）

A. this 关键字指向当前对象自身，super 关键字指向当前对象的直接父类

B. 在 main 方法中可以存在 this 或 super 关键字，但不能同时存在。

C. this 和 super 关键字都可以访问成员属性，成员方法和构造方法

D. 在一个类的构造方法中可以同时使用 this 和 super 来调用其他构造方法

3. 给出下面的代码段，在代码说明//assignment x=a,y=b 处写入如下（ D ）个代码是正确的。（选择一项）

```
public class Base {
    int w, x, y, z;
    public Base(int a, int b) {
        x = a;
        y = b;
    }
    public Base(int a, int b, int c, int d) {
        // assignment x=a,y=b
        w = d;
        z = c;
    }
}
```

A Base(a,b);

B. x=a,y=b;

C. this(a),this(b);

D. this(a,b)

4. 以下 **Java** 程序编译运行后的输出结果是（ A ）。（选择一项）

```
class HelloA {
    public HelloA( ) {
        System.out.println("HelloA");
    }
    {   System.out.println("I'm A class");
    }
    static {
        System.out.println("static A");
    }
}
public class HelloB extends HelloA {
    public HelloB( ) {
        System.out.println("HelloB");
    }
    {   System.out.println("I'm B class");
    }
    static {
        System.out.println("static B");
    }
    public static void main(String[] args) {
        new HelloB( );
    }
}
```

A
```
static A
static B
I'm A class
```

C.
```
static A
I'm A class
HelloA
```

HelloA
I'm B class
HelloB

B.
static A
static B
I'm A class
I'm B class
HelloA
HelloB

D
static B
I'm B class
HelloB

D
static A
static B
HelloA
HelloB
I'm A class
I'm B class

5. **下列选项中关于 Java 中 super 关键字的说法正确的是（ AD）。（选择二项）**

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关键字可以调用父类的构造方法

二、 判断题

1. 如果定义一个类时没有使用 extends，则它的父类是 java.lang.Object。（ T ）
2. 对子类实例化对象需要调用超类的构造函数进行初始化工作，完成对子类中的超类实例变量的初始化，则子类可以继承超类的构造函数。（ F ）
3. 在 Java 中任何一个子类的构造方法都必须调用其父类的构造方法(包括隐式调用)，并且调用父类的构造方法的语句必须是子类构造方法的第一条语句。（ T ）
4. Java 中的类不允许多重继承，一个类只能有有一个直接父类。 T
5. Object 类有一个 public 方法是 toString( )，一个对象通过调用该方法可以获得该对象的字符串表示。（ T ）
6. 父类 Person 中定义了一个 private void show()的方法，那么子类要重写这个方法时，方法的访问修饰符可以是默认的，protected 或 public。（ T ）
7. 运算符“==”用于比较引用时，如果两个引用指向内存同一个对象，则返回 true。（ T ）
8. 构造方法中，第一句话总是 super。（ F ）

三、 简答题

1. 继承的好处是什么? JAVA 的继承使用哪个关键字实现？定义某个类时，如果没有使用继承关键字，那么继承了哪个类?
2. 方法重载和方法重写（覆盖）的区别。
3. Object 类中的 toString 方法能否被子类重写?请做测试。
4. 重写中，子类的返回值类型可不可以跟父类不完全一致?说出你的理由。
5. Object 是所有类的根类吗?是所有类的直接父类吗?在哪里查看 Object 类的源代码?
6. java.lang.Object 类的六个常用方法的声明并说明其作用。
7. 继承条件下子类构造方法的执行过程
8. 假如父类有 main 方法，子类能不能继承?
9. super 关键字的作用和使用
10. ==和 equals()的联系和区别

1　继承的好处是什么? JAVA 的继承使用哪个关键字实现? 定义某个类时，如果没有使用继承关键字，那么继承了哪个类?
为了提高代码的复用性，extends，Object类
2　方法重载和方法重写（覆盖）的区别。
重载是名字相同，形参不同，方法重写是名字，参数类型，参数个数和父类完全相同
3　Object 类中的 toString 方法能否被子类重写?请做测试。
可以
4　重写中，子类的返回值类型可不可以跟父类不完全一致?说出你的理由。
可以，子类的返回值可以与父类相同，也可以是父类方法返回值的子类
5　Object 是所有类的根类吗?是所有类的直接父类吗?在哪里查看 Object 类的源代码?
Object是所有类的根类不是所有类的直接父类，在JDK的src中
6　java.lang.Object 类的六个常用方法的声明并说明其作用。
7　继承条件下子类构造方法的执行过程
用super调用父类的属性
8　假如父类有 main 方法，子类能不能继承?
可以继承
9　super 关键字的作用和使用
调用父类属性
10　==和 equals()的联系和区别
如果在object类中，没区别
再String中，equals重写了方法，比较的是字符串，==比较的是地址

1

```
3 个用法  1 个继承者  新 *
public class Circle {
    8 个用法
    private Double radius;

    2 个用法  新 *
    public Circle() {
        this.radius = 0.0;
    }

    1 个用法  新 *
    public Circle(Double radius) {
        this.radius = radius;
    }

    0 个用法  新 *
    public Double getRadius() {
        return radius;
    }

    2 个用法  新 *
    public void setRadius(Double radius) {
        this.radius = radius;
    }

    2 个用法  新 *
    protected Double getArea() {
        return radius * radius * Math.PI;
    }

    1 个用法  新 *
    protected Double getPerimeter() {
        return 2 * Math.PI * radius;
    }
```

```
public class Cylinder extends Circle{
    5 个用法
    private Double height;

    1 个用法  新 *
    public Cylinder(Double height) {
        super();
        this.height = height;
    }

    0 个用法  新 *
    public Cylinder(Double radius, Double height) {
        super(radius);
        this.height = height;
    }

    0 个用法  新 *
    public Double getHeight() {
        return height;
    }

    0 个用法  新 *
    public void setHeight(Double height) {
        this.height = height;
    }

    1 个用法  新 *
    protected Double getVolume(){
        return super.getArea() * this.height;
    }
```

```
    1 个用法  新 *
    protected Double getVolume(){
        return super.getArea() * this.height;
    }

    2 个用法  新 *
    protected void showVolume(){
        System.out.println("体积为: " + this.getVolume());
    }
}
```

```
    2 个用法  新 *
    protected void show(){
        System.out.println("半径: " + this.radius+ "周长: " + this.getPerimeter()+ "面积: " + this.getArea());
    }
}
```

```
public class CircleAndCylinderLauncher {
    新 *
    public static void main(String[] args) {
        Circle circle = new Circle();
        circle.show();
        Cylinder cylinder = new Cylinder( height: 10.0);
        cylinder.showVolume();
        circle.setRadius(2.0);
        circle.show();
        cylinder.setRadius(3.0);
        cylinder.showVolume();
    }
}
```

四、 编码题

1. 编写应用程序，创建类的对象，分别设置圆的半径、圆柱体的高，计算并分别显示圆半径、圆面积、圆周长，圆柱体的体积。（7 分）

   提示:（1）编写一个圆类 Circle，该类拥有：

   1) 一个成员变量，radius（私有，浮点型）;//存放圆的半径；
   2) 两个构造方法

      Circle（）　　//将半径设为 0

      Circle（double　r ）　　//创建 Circle 对象时将半径初始化为 r
   3) 三个成员方法

      double getArea（）　　//获取圆的面积

      double getPerimeter（）　//获取圆的周长

      void show（）　　//将圆的关径、周长、面积输出到屏幕

   （2） 编写一个圆柱体类 Cylinder，它继承于上面的 Circle 类。还拥有：

   1) 一个成员变量，double hight　（私有，浮点型）；　//圆柱体的高；
   2) 构造方法

      //创建 Cylinder 对象时将半径初始化为 r,高度初始化为 h

      Cylinder（double r,double h）
   3) 成员方法

      double getVolume（）　　//获取圆柱体的体积

      void showVolume（）　　//将圆柱体的体积输出到屏幕

2. 请使用面向对象的思想，设计自定义类，描述出租车和家用轿车的信息。

   设定

   1) 出租车类:
      属性包括：车型，车牌，所属出租公司；方法包括：启动，停止
   2) 家用轿车类：
      属性包括：车型，车牌，车主姓名；方法包括：启动，停止

   要求

   1) 分析出租车和家用轿车的公共成员，提取出父类—汽车类
   2) 利用继承机制，实现出租车类和家用轿车类
   3) 编写测试类，分别测试汽车类，出租车类和家用轿车类对象的相关方法
   4) 定义名为 car 的包存放汽车类，出租车类，家用轿车类和测试类

   运行效果

```java
public class Car{
    3 个用法
    private String vehicleType;
    3 个用法
    private String licencePlate;


    3 个用法  新 *
    public Car(String vehicleType, String licencePlate) {
        this.vehicleType = vehicleType;
        this.licencePlate = licencePlate;
    }

    3 个用法  2 个重写  新 *
    public void run(){
        System.out.println("我是车，我启动");

    }

    3 个用法  2 个重写  新 *
    public void stop(){
        System.out.println("我是车，我停止");
    }

    0 个用法  新 *
    public String getVehicleType() { return vehicleType; }

    0 个用法  新 *
    public void setVehicleType(String vehicleType) { this.vehicleType = vehicleType; }

    1 个用法  新 *
    public String getLicencePlate() { return licencePlate; }

    0 个用法  新 *
    public void setLicencePlate(String licencePlate) { this.licencePlate = licencePlate; }
```

```java
public class TAndSeLauncher {
    新 *
    public static void main(String[] args) {
        Car car = new Car( vehicleType: "",  licencePlate: "");
        car.run();
        car.stop();
        System.out.println("==================================");
        Sedan sedan = new Sedan( vehicleType: "1",  licencePlate: "q",  nameofOwner: "武大朗");
        sedan.run();
        sedan.stop();
        System.out.println("==================================");
        Taxi taxi = new Taxi( vehicleType: "",  licencePlate: "京B123",  ofTaxiCompany: "景顺出租车公司");
        taxi.run();
        taxi.stop();

    }
}
```

```java
public class Sedan extends Car{
    2 个用法
    private String nameofOwner;

    1 个用法  新 *
    public Sedan(String vehicleType, String licencePlate, String nameofOwner) {
        super(vehicleType, licencePlate);
        this.nameofOwner = nameofOwner;
    }

    3 个用法  新 *
    @Override
    public void run() {
        System.out.println("我是".concat(this.nameofOwner).concat( str: ",").concat( str: "我的汽车我做主"));

    }

    3 个用法  新 *
    @Override
    public void stop() {
        System.out.println("目的地到了，我们去玩吧");
    }
}
```

```java
public class Taxi extends Car{
    2 个用法
    private String ofTaxiCompany;

    1 个用法  新 *
    public Taxi(String vehicleType, String licencePlate, String ofTaxiCompany) {
        super(vehicleType, licencePlate);
        this.ofTaxiCompany = ofTaxiCompany;
    }

    3 个用法  新 *
    @Override
    public void run() {
        System.out.println("乘客您好");
        System.out.println("我是".concat(this.ofTaxiCompany).concat("的，我的车牌是".concat(this.getLicencePlate())
                .concat( str: "您要去哪里？")));
    }

    3 个用法  新 *
    @Override
    public void stop() {
        System.out.println("目的地已经到了，请您付费下车，欢迎再次乘坐");
    }
}
```

```
我是车，我启动
我是车，我停止
============================
我是武大郎，我的汽车我做主
目的地到了，我们去玩吧
============================
乘客您好
我是景顺出租车公司的，我的车牌是京B123，您要去哪里？
目的地已经到了，请您付费下车，欢迎再次乘坐
```

3. 某公司要开发新游戏，请用面向对象的思想，设计游戏中的蛇怪和蜈蚣精

设定

1) 蛇怪类:

属性包括：怪物名字，生命值，攻击力

方法包括：攻击，移动（曲线移动），补血（当生命值<10 时，可以补加 20 生命值）

2) 蜈蚣精类：

属性包括：怪物名字，生命值，攻击力

方法包括：攻击，移动（飞行移动）

要求

1) 分析蛇怪和蜈蚣精的公共成员，提取出父类—怪物类

2) 利用继承机制，实现蛇怪类和蜈蚣精类

3) 攻击方法，描述攻击状态。内容包括怪物名字，生命值，攻击力

4) 编写测试类，分别测试蛇怪和蜈蚣精的对象及相关方法

5) 定义名为 mon 的包存怪物类，蛇怪类，蜈蚣精类和测试类

运行效果

```
怪物蛇妖甲展开攻击
当前生命值是:5
攻击力是:20
实施大蛇补血术。。。。。，当前生命值是：25
我是蛇怪，我走S型路线
==========================
怪物蜈蚣乙展开攻击
当前生命值是:60
攻击力是:15
我是蜈蚣精，御风飞行
```

五、 可选题

1. 请用面向对象的思想，设计自定义类描述演员和运动员的信息

设定

1) 演员类:

属性包括：姓名，年龄，性别，毕业院校，代表作

```java
public class Monster {
    3 个用法
    private String name;
    4 个用法
    private Integer bleed;
    4 个用法
    private Integer attack;

    2 个用法  新 *
    public Monster(String name, Integer bleed, Integer attack) {
        this.name = name;
        this.bleed = bleed;
        this.attack = attack;
    }

    2 个用法  2 个重写  新 *
    protected void move(){
    }
    2 个用法  2 个重写  新 *
    protected void attack(){
    }

    2 个用法  新 *
    protected void show(){
        System.out.println("当前声明值: " + this.bleed);
        System.out.println("攻击力是: " + this.attack);
    }

    新 *
    public String getName() {
        return name;
    }

    0 个用法  新 *
    public void setName(String name) {
        this.name = name;
    }

    3 个用法  新 *
    public Integer getBleed() {
        return bleed;
    }

    1 个用法  新 *
    public void setBleed(Integer bleed) {
        this.bleed = bleed;
    }

    0 个用法  新 *
    public Integer getAttack() {
        return attack;
    }

    0 个用法  新 *
    public void setAttack(Integer attack) {
        this.attack = attack;
    }
}
```

```java
public class Scolopendra extends Monster {
    1 个用法  新 *
    public Scolopendra(String name, Integer bleed, Integer attack) {
        super(name, bleed, attack);
    }

    2 个用法  新 *
    @Override
    protected void move() {
        System.out.println("我是".concat(this.getName()).concat( str: ", 御剑飞行"));
    }

    2 个用法  新 *
    @Override
    protected void attack() {
        System.out.println("怪物蜈蚣乙展开攻击");
    }
}
```

```java
public class Snake extends Monster{
    1 个用法  新 *
    public Snake(String name, Integer bleed, Integer attack) {
        super(name, bleed, attack);
    }

    2 个用法  新 *
    @Override
    protected void move() {
        System.out.println("我是".concat(this.getName()).concat( str: ", 我走S型路线"));
    }

    2 个用法  新 *
    @Override
    protected void attack() {
        System.out.println("怪物蛇妖甲展开攻击");
    }

    1 个用法  新 *
    protected void addBlood(){
        if (this.getBleed() < 10){
            Integer temp = this.getBleed() + 20;
            this.setBleed(temp);
            System.out.println("实施大蛇补血术。。。。。，当前生命值是: " + this.getBleed());
        }
    }
}
```

```java
public class TestMonsters {
    新 *
    public static void main(String[] args) {
        Snake snake = new Snake( name: "蛇怪",  bleed: 5,  attack: 20);
        snake.attack();
        snake.show();
        snake.addBlood();
        snake.move();
        System.out.println("====================");

        Scolopendra scolopendra = new Scolopendra( name: "蜈蚣精",  bleed: 60,  attack: 15);
        scolopendra.attack();
        scolopendra.show();
        scolopendra.move();
    }
}
```

方法包括：自我介绍

2) 运动员类：

属性包括：姓名，年龄，性别，运动项目，历史最好成绩

方法包括：自我介始

要求

3) 分析演员和运动员的公共成员，提取出父类—人类

4) 利用继承机制，实现演员类和运动员类

5) 编写测试类，分别测试人类，演员类和运动员类对象及相关方法

6) 定义名为 act 的包存人类，演员类，运动员类和测试类

运行效果