吉软 java 基础第一周测试

【金城君组】【蔡俊豪】

说明：

1. 上方【组】填入所在的组，上方【姓名】填入自己的真实姓名。
2. 答题方式，基于 Word 文档基础上答题
3. 编程题可利用工具编程完以后，复制到该文档内。
4. 答完以后，导成 PDF。以姓名.PDF 命名。上传至老师指定邮箱。

一、 选择题（共 10 题，每题 2 分）

1、以下代码运行输出是(C)

```
public class Person{

    private String name="Person";
        int age=0;
    }

public class Child extends Person{

    public String grade;

    public static void main(String[] args){
        Person p = new Child();
        System.out.println(p.name);

    }
}
```

A) 输出：Person

B) 没有输出

C) 编译出错

D) 运行出错

2、在使用 super 和 this 关键字时，以下描述正确的是(A)

A) 在子类构造方法中使用 super()显示调用父类的构造方法，super()必须写在子类构造方法的第一行，否则编译不通过

B) super()和 this()不一定要放在构造方法内第一行

C) this()和 super()可以同时出现在一个构造函数中

D) this()和 super()可以在 static 环境中使用，包括 static 方法和 static 语句块

3、以下对封装的描述正确的是(D)

A) 只能对一个类中的方法进行封装，不能对属性进行封装

B) 如果子类继承了父类，对于父类中进行封装的方法，子类仍然可以直接调用

C) 封装的意义不大，因此在编码时尽量不要使用

D) 封装的主要作用在于对外隐藏内部实现细节，增强程序的安全性

4. 以下对继承的描述错误的是(A)

A) Java 中的继承允许一个子类继承多个父类

B) 父类更具有通用性，子类更具体

C) Java 中的继承存在着传递性

D) 当实例化子类时会递归调用父类中的构造方法

5、以下程序的运行结果是(D)

```
class Person{
    public Person(){
        System.out.println(“this is a Person”);
    }
}

public class Teacher extends Person{
    private String name=”tom”;

    public Teacher(){
        System.out.println(“this is a teacher”);
        super();
    }

     public static void main(String[] args){
        Teacher teacher = new Teacher();
         System.out.println(this.name);

    }

}
```

A) this is a Person
this is a teacher
tom

B) this is a teacher
this is a Person
tom

C) 运行出错

D) 编译有两处错误

6、以下说法错误的是(D)

A) super.方法()可以调用父类的所有非私有方法

B) super()可以调用父类的所有非私有构造函数

C) super.属性可以调用父类的所有非私有属性

D) this 和 super 关键字可以出现在同一个构造函数中

7、访问修饰符作用范围由大到小是(B)

A) private-default-protected-public

B) public-default-protected-private

C) private-protected-default-public

D) public-protected-default-private

8、以下(D)不是 Object 类的方法

A) clone()

B) finalize()

C) toString()

D) hasNext()

9、以下对重载描述错误的是(B)

A) 方法重载只能发生在一个类的内部

B) 构造方法不能重载

C) 重载要求方法名相同，参数列表不同

D) 方法的返回值类型不是区分方法重载的条件

10、 以下对接口描述错误的有(D)

A) 接口没有提供构造方法

B) 接口中的方法默认使用 public、abstract 修饰

C) 接口中的属性默认使用 public、static、final 修饰

D) 接口不允许多继承

二、 填空题（共 10 题，每题 2 分）

1. 如果一个方法不返回任何值，则该方法的返回值类型为 void 。

2. 如果子类中的某个 方法名 、 返回参数 和 形参 与父类中的某个方法完全一致，则称子类中的这个方法覆盖了父类的同名方法。

3. 接口中所有的属性均为 public 、 static 和 final 的。

4、局部变量的名字与成员变量的名字相同，若想在该方法内使用成员变量，必须使用关键字 this

5、 java.lang 包是 java 语言的核心包，它包含了运行 java 程序必不可少的系统类，在使用这个包中的类时不需要 import。

6、对于 int 型变量，内存分配 4 个字节。

7、在 Java 中有二维数组 int [ ] [ ] array={{1,2,3},{4,5}}，可以使用 array[0].length 得到二维数组中第二维中第一个数组的长度。

8、 java 命令用于执行在 Java 虚拟机中运行类的类文件。

9、Java 提供了三种注释类型，分别是 单行注释 ， 多行注释 和 文档注释 。

10、基本数据类型的类型转换中，要将 double 类型的常量 3.14159 赋给为整数类型变量 n 的语句是 int n = (int)3.14159; 。

三、 简答题（共 5 题，每题 4 分）

一、 如果有两个类 A、B(注意不是接口)，你想同时使用这两个类的功能，那么你会如何编写这个 C 类呢?

用 B 类继承 A 类，再用 C 类继承 B 类，因为 JAVA 无法实现多继承。

二、 谈谈你对抽象类和接口的理解。

抽象类可以拥有非抽象方法，抽象类是被 abstract 关键字修饰的类，还可以定义变量，接口里面必须都是抽象方法以及他的变量都会被默认修饰为常量。抽象类：is B；接口类：has B 。

三、 静态变量和实例变量的区别。

静态变量需要关键字 **static**，类被加载时便直接开辟空间；实例变量不需要 static，实例在创建对象时会分配空间。静态变量可以根据类名来直接访问，但是实例变量不可以。

四、 **Overload** 和 **Override** 的区别。**Overload** 是否可以改变返回值

Overload（方法的重载）：在同一个类中，方法名不变，形参改变，除此之外可以改变返回值，与返回值类型无关。**Override**（方法的重写）：方法名，形参都不变，是子类对父类的方法进行重写。

五、 **swtich** 是否能作用在 **byte** 上，是否能作用在 **long** 上，是否能作用在 **String** 上?

Switch 可以作用在 **byte**，**String** 上，不能作用在 **long** 上。

六、 编程题（共 3 题，第（1）(2）题 10 分，第（3）题 20 分)

1、从控制台输入电话号，判断是固定电话（8 位）还是手机号（11 位）。

- **a)** 如果是固定电话，则把其奇数位分别加 **5** 后，打印输出新的电话号。
- **b)** 如果是手机号，则把偶数位分别加 **8** 后，打印输出新的手机号。
- **c)** 要求计算后的电话号码位数不变

```
package Exam.Exam01;

import java.util.Scanner;

/**
 * @author junhaocai
 * @email junhaocai01@gmail.com
 * @date 2023/1/8
 */
public class Telephone {
    public static void main(String[] args) {
        System.out.println("请输入电话号码： ");
        Scanner scanner = new Scanner(System.in);
        String num = scanner.next();
        int [] nums = new int[num.length()];
        if (num.length() == 11){
            for (int i = 0; i < 11; i++) {
                if (i % 2 == 1){
                    nums[i] =   Integer.parseInt(String.valueOf(num.charAt(i))) +
```

```
8;
                        if (nums[i] / 10 %10 != 0) {
                            nums[i] = nums[i] % 10;
                        }
                    }
                    else {
                        nums[i] =   Integer.parseInt(String.valueOf(num.charAt(i)));
                    }

                }
                for (int i : nums) {
                    System.out.print(i);
                }
            }
            if (num.length() == 8){
                for (int i = 0; i < 8; i++) {
                    if (i % 2 == 0){
                        nums[i] =   Integer.parseInt(String.valueOf(num.charAt(i))) +
5;
                        if (nums[i] / 10 %10 != 0) {
                            nums[i] = nums[i] % 10;
                        }

                    }
                    else {
                        nums[i] =   Integer.parseInt(String.valueOf(num.charAt(i)));

                    }
                }
                for (int i : nums) {
                    System.out.print(i);
                }
            }
            else {
                System.out.println("输入有误");
            }
        }
}
```

2、 创建电子产品类，包含存储数据的方法。

- **a) 创建电脑类，可以存储数据以及玩游戏；**
- **b) 创建手机类，可以存储数据以及打电话；**
- **c) 自行设计题目的结构（比如：接口，抽象类，继承，实现等…）**

```java
package Exam.Exam02;

/**
 * @author junhaocai
 * @email junhaocai01@gmail.com
 * @date 2023/1/8
 */
public class Computer extends Product implements PlayGame {
    @Override
    public void storeMemory() {
        System.out.println("电脑已存储数据!  ");
    }

    public void playGame(){
        System.out.println("玩游戏");
    }
}
```

```java
package Exam.Exam02;

/**
 * @author junhaocai
 * @email junhaocai01@gmail.com
 * @date 2023/1/8
 */
public abstract class Product {
    public abstract void storeMemory();
}
```

```java
package Exam.Exam02;

/**
 * @author junhaocai
 * @email junhaocai01@gmail.com
 * @date 2023/1/8
 */
public class Telephone extends Product implements CallUp{

    @Override
    public void storeMemory() {
        System.out.println("手机已存储数据!  ");
    }

    public void callUp(){
```

```
            System.out.println("打电话!  ");
        }
}
```

```
package Exam.Exam02;

public interface CallUp {
        void callUp();
}
```

```
package Exam.Exam02;

public interface PlayGame {
        void playGame();
}
```

3、创建 Animal 类，创建 Dog 类以及 Cat 类，分别继承（或实现）Animal 类。Dog 和 Cat 分别有不同的 eat 方法。
再创建 Food 类，创建 Bone 类以及 Fish 类，分别继承（或实现）Food 类，Bone 和 Fish 有不同的属性。
创建 Test 类，分别测试 Dog 吃 Bone，以及 Cat 吃 Fish。

```
package Exam.Exam03;

/**
 * @author junhaocai
 * @email junhaocai01@gmail.com
 * @date 2023/1/8
 */
public abstract class Animal {
    public abstract void eat(Object obj);
}
```

```
package Exam.Exam03;

/**
 * @author junhaocai
 * @email junhaocai01@gmail.com
 * @date 2023/1/8
 */
public class Dog extends Animal{
        @Override
        public void eat(Object obj) {
            ((Bone) obj).beEaten();
```

```
    }
}
package Exam.Exam03;

/**
 * @author junhaocai
 * @email junhaocai01@gmail.com
 * @date 2023/1/8
 */
public class Cat extends Animal{

    @Override
    public void eat(Object obj) {
        ((Fish) obj).beEaten();
    }
}
package Exam.Exam03;/**
*     @author junhaocai
*     @email junhaocai01@gmail.com
*     @date 2023/1/8
*/
public abstract class Food {
    public abstract void beEaten();
}
package Exam.Exam03;

/**
 * @author junhaocai
 * @email junhaocai01@gmail.com
 * @date 2023/1/8
 */
public class Fish extends Food{
    public void beEaten(){
        System.out.println("鱼被吃了！  ");
    }
}
package Exam.Exam03;

/**
 * @author junhaocai
 * @email junhaocai01@gmail.com
 * @date 2023/1/8
 */
public class Bone extends Food{
```

```
    @Override
    public void beEaten() {
        System.out.println("骨头被吃了");
    }
}
package Exam.Exam03;

/**
 * @author junhaocai
 * @email junhaocai01@gmail.com
 * @date 2023/1/8
 */
public class Test {
    public static void main(String[] args) {
        Dog dog = new Dog();
        Object bone = new Bone();
        dog.eat(bone);

        Cat cat = new Cat();
        Fish fish = new Fish();
        cat.eat(fish);
    }
}
```