周测

一、选择题((共15题，每题3分，共45分)

1) (多选)在Java中，下列关于方法重载的说法中错误的是（cd）。

A. 方法重载要求方法名称必须相同

B. 重载方法的参数列表必须不一致

C. 重载方法的返回类型必须一致

D. 一个方法在所属的类中只能被重载一次

2)(单选) 在Java中，下面对于构造函数的描述正确的是（b）。

   A. 类必须显式定义构造函数

 B. 构造函数的返回类型是void

 C. 构造函数和类有相同的名称，并且不能带任何参数

 D. 一个类可以定义多个构造函数

3) (单选)下面Java代码的运行结果是（C）。

 class Penguin {

  private String name=null; // 名字

private int health=0; // 健康值

private String sex=null; // 性别

  public void Penguin() {

health = 10;

sex = "雄";

System.out.println("执行方法。");

}

public void print() {

System.out.println("企鹅的名字是"

 + name +  "，健康值是" + health + "，性别" + sex+ "。");

}

public static void main(String[] args) {

Penguin pgn = new Penguin();

pgn.print();

}

}

  A. 企鹅的名字是null，健康值是10，性别是雄。

 B. 执行方法。企鹅的名字是null，健康值是0，性别是null。

C. 企鹅的名字是null，健康值是0，性别是null。

D. 执行构造方法。企鹅的名字是null，健康值是10，性别是雄。

4) (单选)下列选项中关于Java中封装的说法错误的是（d ）。

A. 封装是将属性私有化，提供公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. 类的属性必须进行封装，否则无法通过编译

5) (单选)下列代码的输出结果是：（C ）

abstract class Vehicle{

public int speed(){

return 0;

}

}

class Car extends Vehicle{

public int speed(){

return 60;

}

}

class RaceCar extends Car{

public int speed(){

return 150;

}

}

public class TestCar {

public static void main(String[] args) {

RaceCar racer = new RaceCar();

Car car = new RaceCar();

Vehicle vehicle = new RaceCar();

System.out.print(racer.speed()+","+car.speed()+","+vehicle.speed());

}

}

A.0,0,0

B.150，60,0

C.150,150,150

D.抛出运行时异常

 6) 在Java中，以下程序编译运行后的输出结果为（ D）。

 public class Test {

     int x, y;

     Test(int x, int y) {

        this.x = x;

        this.y = y;

     }

     public static void main(String[] args) {

        Test pt1, pt2;

        pt1 = new Test(3, 3);

        pt2 = new Test(4, 4);

        System.out.print(pt1.x + pt2.x);

     }  }

A. 6  B. 3 4  C. 8  D. 7

 7)(单选) 在Java语言中，下列关于类的继承的描述，正确的是（B）。

A. 一个类可以继承多个父类

B. 一个类可以具有多个子类

C. 子类可以使用父类的所有方法

D. 子类可以继承父类构造方法

 8) (单选)给定一个Java程序Test.java的代码如下所示，编译时，会出现以下（A）情况。

 class Parent {

 public int count() {               // 第1行

return 0;

 }

}

public class Test  extends Parent {

 private int i;

  public int count() {               // 第2行

return i % 9;                 // 第3行

}

}

A. 编译通过

B. 在第1行引发编译错误

C. 在第2行引发编译错误

D. 在第3行引发编译错误

9) (单选)给定如下一个Java源文件Child.java，编译并运行Child.java，以下结果正确的是（B）。

 class Parent1 {

    Parent1(String s){

        System.out.println(s);

    }

}

class Parent2  extends Parent1{

    Parent2(){

        System.out.println("parent2");

    }

}

public class Child extends Parent2 {

    public static void main(String[] args) {

        Child child = new Child();

    }

}

A. 编译错误：没有找到构造器Child() {}

  B. 编译错误：没有找到构造器Parent1(){}

  C. 正确运行，没有输出值

  D. 正确运行，输出结果为：parent2

 10) (单选)请看下列代码:

public abstract class Shape{

int x;

int y;

public abstract void draw();

public void setAnchor(int x,int y){

this.x=x;

this.y=y;

}

}

下列选项中能正确使用Shape类的是( c)

A.public class Circle implements Shape{

private int radius; }

B.public abstract class Circle implements Shape{

private int radius;

public void draw();

}

C.public class Circle extends Shape{

private int radius;

public void draw();

}

D.public class Circle extends Shape{

private int radius;

public void draw(){/\*code here \*/}

}

 11) (多选)在Java语言中，下列说法正确的是（Acd）。

A. 一个接口可以继承多个接口

B. 一个类可以继承多个类

C. 一个类可以实现多个接口

D. 一个类可以有多个子类

 12)(单选) 下面关于Java接口的说法错误的是（c）。

  A. 一个Java接口是一些方法特征的集合，但没有方法的实现

  B. Java接口中定义的方法在不同的地方被实现，可以具有完全不同的行为

  C. Java接口中可以声明私有成员

  D. Java接口不能被实例化

 13) (多选)在Java接口中，下列选项中属于有效的方法声明是（ab）。

  A. public void aMethod();

  B. void aMethod();

  C. void aMethod(){}

  D. private void aMethod();

14)在java 中,下列( ) 类不能派生出子类. ( d ) (单选)

A. public class MyClass{ }

B. class MyClass{ }

C. abstract class MyClass{ }

D. final class MyClass { }

15)在Java中，下面关于抽象类的描述正确的是（bd）。（多选）

A. 抽象类可以被实例化

B. 如果一个类中有一个方法被声明为抽象的，那么这个类必须是抽象类

C. 抽象类中的方法必须都是抽象的

D. 声明抽象类必须带有关键字abstract

三、程序改错题（10分）

说明： 下述程序中都包含着错误而不能通过编译。请在右栏指出程序错误的原因（可指明行号），并修改程序使其可以通过编译。

程序：

01 abstract class Animal {

02 public abstract void saySomething() //抽象方法没有方法体所以删除方法体。

03

05 }

06 class Dog extends Animal {

07 public void saySomething() {

08 System.out.println("我现在有了第二职业：捉耗子！");

09 }

10 }

11 public class Test {

12 public static void main (String[] args) {

13 Animal ani=new Dog();

14 ani.saySomething();

15 }

16 }

三、画图题(每题10分,共20分)

1.请根据如下相关Emp及Test类的代码画图对应的内存管理图并写出运行结果

class Emp {

public int age;

}

public class Test {

public static void change(Emp emp){

emp.age = 50;

emp = new Emp();//再创建一个对象

emp.age=100; }

public static void main(String[] args) {

Emp emp = new Emp();

emp.age = 100;

System.out.println(emp.age);

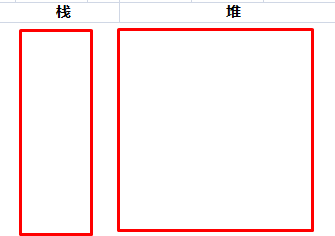
change(emp);

System.out.println(emp.age);

}

}

输出结果为: \_100 50\_\_\_\_\_\_\_\_\_



Change

main

Age=100

Age=50

new Emp()

2.请根据如下相关Emp及Test类的代码画图对应的内存管理图并写出运行结果

public class Test6 {

public static void main(String[] args) {

Child c=new Child();

}}

class Parent{

private int a=2;

public Parent(){

System.out.println("Parent:"+a);

show();}

public void show(){

System.out.println("a1:"+a);

}}

class Child extends Parent{

int a=3;

public Child(){

System.out.println("Child:"+a);}

public void show(){

System.out.println("a2:"+a);

}

}

输出结果为: Parent:2 a2:0 Child:3

三、程序题：（共25分）

按要求完成一下代码：

1. 定义接口Swim： --- 3分

抽象方法：swimming()

2. 定义接口Fly: --3分

抽象方法：flying()

3. 定义类Bird： --- 3分

方法： layegg(){输出“鸟会下蛋”}

4. 定义类Penguin(企鹅)：会游泳 --3分

1. 继承Bird类

2. 实现Swim接口，重写swimming(),输出“企鹅会游泳”

5. 定义类Sparrow（麻雀）：会飞 --3分

1. 继承Bird类

2. 实现Fly接口，重写flying(),输出“麻雀会飞”

6. 定义类Eagle(老鹰)：会飞 -- 3分

1. 继承Bird类

2. 实现Fly接口，重写flying(),输出“老鹰会飞”

7. 创建Test类，main方法中： --7分

1. 创建Bird[],长度为5,向数组中存入2只企鹅，2只麻雀，1只老鹰

2.遍历数组，若遍历到的对象是麻雀或老鹰，造型成Fly类型，调用flying（）；

若遍历到的对象是企鹅，造型成Swim类型，调用swimming()

**package** qwe;

**public** **class** AABB {

**public** **interface** Swim {

**void** swimming();

}

**public** **interface** Fly {

**void** flying();

}

**public** **class** Bird {

**public** **void** layegg() {

System.***out***.println("鸟会下蛋");

}

}

**public** **class** Penguin **extends** Bird **implements** Swim {

**public** **void** swimming() {

System.***out***.println("企鹅会游泳");

}

}

**public** **class** Sparrow **extends** Bird **implements** Fly {

**public** **void** flying() {

System.***out***.println("麻雀会飞");

}

}

**public** **class** Eagle **extends** Bird **implements** Fly {

**public** **void** flying() {

System.***out***.println("老鹰会飞");

}

}

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Bird[] birds = **new** Bird[5];

birds[0] = **new** Penguin();

birds[1] = **new** Penguin();

birds[2] = **new** Sparrow();

birds[3] = **new** Sparrow();

birds[4] = **new** Eagle();

**for**(**int** i=0;i<birds.length;i++) {

Bird bs = birds[i];

**if**(bs != Fly) {

fly.flying();

}**else** {

swim.swimming();

}

}

}

}

}