第四组

一、选择题

1.参考：https://www.runoob.com/java/java-modifier-types.html

关键点总结：

default 使用对象：类、接口、变量、方法。

private 使用对象：变量、方法。 注意：不能修饰类（外部类）

public 使用对象：类、接口、变量、方法

protected 使用对象：变量、方法。 注意：不能修饰类（外部类）。

private 和 protected 可以修饰内部类。

static 使用对象：成员变量(类变量)、方法(类方法)、内部类

final 使用对象：类、方法、变量

abstract 使用对象：类、方法

方法中的变量，即局部变量，不能用 public、protected、private 以及 static 修饰符

访问控制					
修饰符	当前类	同一包内	子孙类(同一包)	子孙类(不同包)	其他包
public	Y	Y	Y	Y	Y
protected	Y	Y	Y	Y/N (说明)	N
default	Y	Y	Y	N	N
private	Y	N	N	N	N

内部类介绍：

https://www.runoob.com/java/java-inner-class.html

https://www.runoob.com/w3cnote/java-inner-class-intro.html(此链接中有关深入理解内部类第二个问题的说明应该

是错的)

静态内部类中，注意以下代码是合法的：

```
class OuterClass {
     static int x = 10;

     static class InnerClass {
       public int myInnerMethod() {
         x=x+10;
         return x;
       }
     }
}

public class test1 {
     public static void main(String[] args) {
       OuterClass.InnerClass myInner = new
OuterClass.InnerClass();
       System.out.println(myInner.myInnerMethod());
     }
}
```

另外，下面的代码是非法的：`class OuterClass {`

```
     static int x = 10;
```

```
    static class InnerClass {
      public int myInnerMethod() {
        x=x+10;
        return x;
      }
    }
}

public class test1 {
    public static void main(String[] args) {
      OuterClass myOuter = new OuterClass();
      OuterClass.InnerClass myInner=myOuter.new
InnerClass();//该句报错 因为内部类是static的
      System.out.println(myInner.myInnerMethod());
    }
}
```

protected 关键字详解(参考用，好像讲的不是太对):

https://www.runoob.com/w3cnote/java-protected-keyword-detailed-explanation.html

protected 关键字和 clone 方法的讲解参见 On Java 基础版和进阶版

Java 接口介绍:

https://www.runoob.com/java/java-interfaces.html

default 关键字介绍(Java 8 默认方法):

https://www.runoob.com/java/java8-default-methods.html

接口里的变量都隐式声明为 public static final,而接口里的方法默认情况下访问权限为 public。

接口及接口的成员变量和成员方法不能声明为 protected、private

2. java.lang 包提供了 Java 语言进行程序设计的基础类，它是默认导入的包，不用导入，就可以直接使用。该包里面的 Runnable 接口和 Object、Math、String、StringBuffer、System、Thread 以及 Throwable 类需要重点掌握，因为它们应用很广。

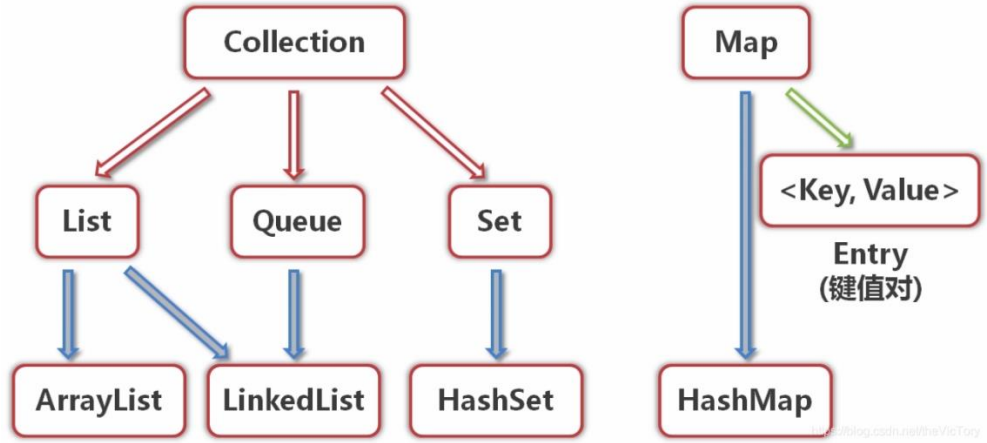
参考：https://www.cnblogs.com/galaxyr/p/15996905.html

5.数据类型有基本数据类型和复合(引用)数据类型：基本数据类型指 boolean,char,byte,short,int,long,float,double 八种，复合数据类型类型有类、接口，典型地，以下几种都属于复合数据类型：字符串 String,基本类型对应的包装类(如下表所示):

基本类型	对应的包装类
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

https://blog.csdn.net/theVicTory

容器(如下图所示):



以上复合数据类型，实际上都是类这一复合数据类型的具体例子。

6.C 项是很微妙的，首先理解默认构造函数的含义：

没有参数的构造函数称为默认构造函数。

另外参考：https://www.yiibai.com/java/constructor.html

注意：如果用户自己没有定义构造函数，则编译器自动生成一个默认构造函数；若用户自定义了构造函数，那么编译器不会生成默认构造函数，也就是说在这种情况下可用的构造函数只有用户自定义的构造函数。

可以结合下面的例子理解：

```
public class test1 {
   int b;

   test1(int a){
      b=a;
   }
```

```
    public static void main(String[] args)  {
        test1 t1=new test1();//编译不通过，提示没有该
构造函数
        System.out.println(t1.b);
    }
}
```

7.D 项十分容易混淆，应记住以下几个知识点：

(1)内部类中有 static 成员时，内部类也必须声明为 static

(2)其他类型的内部类中(即除 static 内部类以外的内部类)不能有静态方法和属性

(3)其他类型的内部类，不能有 static 类型属性，却可以有常量。即：

```
public class Outer{
    int x;
    class Inner{
        static int a = 0;//这样写是不合法的
        static final int b=0;//这样写是合法的
         int c=0;//合法的
    }
}
```

二、填空题

11.作为对比，注意：静态内部类，即 static 内部类中既可以声明 static 成员，也可以声明非 static 成员；而其它内部类只能声明非 static 成员。有关内部类这方面的例子参考选择题第 7 题，现在给出本题的例子以加深理解。

例一：

```
public class test1 {
   int x;
   void c() {}

    public static void main(String[] args)  {
        System.out.println(x);//报错
        c();//报错
    }
}
```

例二：

```
public class test1 {
   static int x;
   static void c() {}
    //正常编译、正常运行
    public static void main(String[] args)  {
        System.out.println(x);
        c();
```

```
    }
}
```

例三：

```
public class test1 {
   int x;
   void c() {}
   //正常编译、正常运行
    public static void main(String[] args)  {
      test1 t=new test1();
         System.out.println(t.x);
         t.c();
    }
}
```

例四：

```
//正常编译运行
public class test1 {
   int x;
   static int y;
   static void c() {}
   void e() {}

   void d() {
```

```
        System.out.println(x);
        System.out.println(y);
        c();
        e();
    }

     public static void main(String[] args)  {

     }
}
```

例五:

```
public class test1 {
    int x;
    static int y;
    static void c() {}
    void e() {}

    static void d() {
        System.out.println(x);//报错
        System.out.println(y);
        c();
        e();//报错
```

```
    }

     public static void main(String[] args)  {


     }
}
```

12.利用类名调用的示例：

```
public class test1 {
   int x;
   static void c() {}

     public static void main(String[] args)  {
       test1 t=new test1();
          System.out.println(t.x);
          test1.c();//使用类名调用静态方法
     }
}
```

三、判断题

3.实例方法就是非 static 类型的方法，就是非静态方法，没有 static 修饰的方法；类方法就是静态方法，使用 static 修饰的方法。

Java 支持的变量类型有：

类变量：独立于方法之外的变量，用 static 修饰。

实例变量：独立于方法之外的变量，不过没有 static 修饰。

局部变量：类的方法中的变量。

知道了以上内容，再结合填空题 11 题的知识点即可解决此题。

5、6 题：

例子：

```
public class Test1 {
   private int a;
    private int b;

    public void setA(int a) {
        this.a = a;
    }
    public void setB(int b) {
        this.b = b;
    }
    public static void main(String[] args)  {
      Test1 t1 = new Test1();
        t1.setA(1);
        t1.setB(2);
        Test1 t2 = t1;//对象赋值
        t2.setA(-1);
```

```
        System.out.println(t1.a);//-1
        System.out.println(t1.b);//2
        System.out.println(t2.a);//-1
        System.out.println(t2.b);//2
    }
}
```

5.t1和t2中实际上是引用，引用里存放的是对象的地址。

6.对象赋值实际上是同一个对象具有两个不同的名字，它们都有同一个引用值，因此上面代码的输出为-1 2 -1 2

7.示例:

```
public class Test1 {
   int a;
    int b;
    //对象数组作方法参数
    public void change(Test1[] t,int i) {
     t[i].a=i;
     t[i].b=i+1;
    }

    public static void main(String[] args)  {
        //为对象数组分配空间
        Test1[] test1=new Test1[5];
```

```
        for(int i=0;i<test1.length;i++) {
        //为各对象本身分配空间，注意没有这句会报空指针
异常
        test1[i]=new Test1();
        System.out.println(test1[i].a+"
"+test1[i].b);
        }
        for(int i=0;i<test1.length;i++) {
        test1[i].change(test1, i);
        System.out.println(test1[i].a+"
"+test1[i].b);//通过该句输出可知形参传递的是引用(指
针)
        }
    }
}
```

输出：

```
0 0
0 0
0 0
0 0
0 0
0 1
```

1 2

2 3

3 4

4 5