一、填空题

1. 语句 Class clz = null；的含义是__Class 类的对象 clz 为 null_____。

2. 给定下列类的定义：

```
class GeometricObject {}
class PolyGon extends GeometricObject {}
class Rectangle extends PolyGon {}
GeometricObject o = new Rectangle ();
Class clz1 = o. getClass ();
```

(1) 声明一个指向 PolyGon 及其子类的类型信息的引用变量 clz 的语句应该是

_____Class<? Extends PolyGon> clz;_____ ；

(2) System.out.println(o.getClass().getSimpleName());的输出结果是___ Rectangle ____;

(3) 下列语句中有错误的是_____2、3_____;

```
Class<PolyGon> clz3 = null;
clz3 = PolyGon.class;          ①
clz3 = Rectangle.class;        ②

Class<? extends PolyGon> clz4 = null;
clz4 = GeometricObject.class;  ③
clz4 = PolyGon.class;          ④
clz4 = Rectangle.class;        ⑤
```

错误原因是（按错误题号解释）_对于 2 语句，clz3 只能指向 PolyGon 的类字面量，所以不能等于 Rectangle.class。对于 3 语句，clz4 只能指向 PolyGon 及其子类的类字面量，而 GeometricObject 是 PolyGon 的父类，所以赋值语句错误。____。

3. 下面五条语句中，错误的有____（2）__（3）_____。

（1）ArrayList<String> lists = new ArrayList<String>();

（2）ArrayList<Object> lists = new ArrayList<String>();

（3）ArrayList<String> lists = new ArrayList<Object>();

（4）ArrayList<String> lists = new ArrayList();

（5）ArrayList lists = new ArrayList<String>();

错误原因是____泛型类型实参并没有继承关系，所以（2）语句中不能把 ArrayList<String>赋给声明类型为 ArrayList<Object>。（3）语句中声明类型和 new 的类型不一致，所以也错误

_____。

泛型没有协变性

使用泛型通配符?将错误的语句修改正确的方法是_____

_（２）改为 ArrayList<?> lists = new ArrayList<String>();

（２）ArrayList<? extends Object> list2 = new ArrayList<String>();

（３）改为 ArrayList<? super String> lists = new ArrayList<Object>();

_____。

4. 下面代码给出了泛型类和非泛型类的定义：

```
class Holder<T> {
    T value;
    public Holder (T value) {this.value = value;}
    public T getValue () {return value;}
}

class RawHolder {
    Object value;
    public RawHolder (Object value) {this.value = value;}
    public Object getValue () {return value;}
}
```

基于上面二个类的定义，有下面四段代码：

① Holder<String> h1 = new Holder<>("aaa"); String s1 = h1. getValue (); System.out.println(s1);	② RawHolder h1 = new RawHolder("aaa"); String s1 = (String)h1. getValue (); System.out.println(s1);
③ Holder<String> h1 = new Holder<> (Integer.valueOf(111)); String s1 = h1. getValue (); System.out.println(s1);	④ RawHolder h1 = new RawHolder (Integer.valueOf(111)); String s1 = (String)h1. getValue (); System.out.println(s1);

上面四段代码中编译通过运行不出错的是_____1、2_____，上面四段代码中编译通过运行出错的是___4___，原因是_ RawHolder 类中 value 为 Object 类型，getValue()返回的类型也为 Object，而编译时由于强制类型转换所以可以通过编译，但是在运行时此时返回的类型为 Integer，但是无法转换为 String，所以会运行出错_____，

上面四段代码中编译不通过是___3___，原因是_ h1. getValue ()返回的类型是 Interger，但是 s1 的声明类型是 String，所以无法通过编译__，

这个例子说明泛型的作用是__把类型的检查在编译中就可以检查出来，不用等到运行时发现类型不匹配再抛出异常_____。

二、单项选择题

1. 泛型参数<T>代表的是_____D_____。

A. 任意类型

B. 某类型的子类型

C. 某类型的父类型

D. 固定指代某种类型

2. 泛型通配符<?>代表的是___A_____。

A. 任意类型

B. 某类型的子类型

C. 某类型的父类型

D. 固定指代某种类型

3. 下面泛型定义中不正确的是_____D_____。

A. class Test1<T> {}

B. interface Test2<T> {}

C. class Test3{<T> void test () {}} //

D. class Test4{void <T> test () {}}

4. 泛型通配符<? extends T>代表的是__B_____。

A. 任意类型

B. 某类型 T 的子类型

C. 某类型 T 的父类型

D. 固定指代某种类型

5. 泛型通配符<? super T>代表的是____C_____。

A. 任意类型

B. 某类型 T 的子类型

C. 某类型 T 的父类型

D. 固定指代某种类型

6. 关于下面代码，描述正确的是____C_____。

```
List<String> list = new ArrayList<String>();
list.add("test");
list.add("red");
list.add (100);
System.out.println(list. size ());
```

A. 输出 2

B. 输出 3

C. 编译错误

D. 运行时报异常

7. 关于下面代码，描述正确的是____B_____。

```
List<Integer> ex_int= new ArrayList<Integer> ();
List<Number> ex_num = ex_int;
System.out.println(ex_num. size ());
```

A. 0

B. 编译错误

C. 运行时报异常

D. 1

8. 下列语句编译时不出错的是_____B_____。

A. List<?> c1 = new ArrayList<String> (); c1.add (new Object ());

B. List<?> c2 = new ArrayList<String> (); c2.add (new String ("1"));

C. List<?> c3 = new ArrayList<String> (); c3.add ("1");

D. List<?> c4 = new ArrayList<String> (); c4.add(null);

9. 给定下列代码：?

```
class Shape {}
class Circle extends Shape {}
class Triangle extends Shape {}
public class Test2_9 {
    public static void main (String [] args) {
        List<? extends Shape> list1 = new ArrayList< Triangle> ();
        List<? extends Shape> list2 = new ArrayList<Circle> ();
```

```
        System.out.println(list1 instanceof List< Triangle>);          ①
        System.out.println(list2 instanceof List);                     ②
        System.out.println(list1.getClass() == list2.getClass());      ③
    }
}
```

则关于语句①②③说法正确的是：_____D_____。

A. ①②③输出结果为 true、 false、 false

B. ①②③输出结果为 true、 true、 true

C. ①编译出错，②③输出结果为 false、 false

D. ①编译出错，②③输出结果为 true、 true

三、多项选择题（一个或多个正确选项）

1. 对于泛型类 class A<T> { … }，T 在 A 类里可以用作不同的地方，在 A 类类体内，下面语句正确的有___A C D F G_____。**ABDG**

A. T x;
B. T m1() {return null;}
C. static T y;
D. void m2(T i) {}
E. static T s1() {return null;}
F. static void s2(T i) {}
G. static <T1> void s3(T1 i, T1 j){}

2. 下列语句编译时不出错的是___A E G H _____。

A. List<? super Integer> x1 = new ArrayList<Number> ();
B. List<? super Number> x2 = new ArrayList<Integer> ();
C. List<? super Number> x3 = new ArrayList<Short> ();
D. List<? super Integer> x4 = new ArrayList<Short> ();
E. List<? extends Number> x5 = new ArrayList<Integer> ();
F. List<? extends Number> x6 = new ArrayList<Object> ();
G. List<Number> x7 = new ArrayList<> ();
H. List<? extends Comparable<Double>> x8 = new ArrayList<Double> ();
I. List<? extends Number> x9 = new ArrayList<int> ();

3. 下面泛型类是 List<?>的子类的是_____A B C_____。

A. List<String>
B. List<Object>
C. List<Integer>
D. List<float>

4. 泛型参数应该写在的位置是_____B D_____。

A. 类名前

B. 类名后

C. 方法名前

D. 方法返回值类型前

5. 关于 java 泛型，下面描述正确的是_____B C D_____。 **ABCD**

A. 泛型的类型参数只能是类类型（包括自定义类），不能是基本类型

B. 泛型的类型参数可以有多个

C. 不能对泛型的具体实例类型使用 instanceof 操作，如 o instanceof ArrayList<String>，否则编译时会出错。

D. 不能创建一个泛型的具体实例类型的数组，如 new ArrayList<String>[10]，否则编译时会出错。

6. 给定下列类和泛型方法的定义：

```
class A {}
class B extends A {}
class C extends B {}
class D extends C {}
public class Test2_9{
    public static <T> void m (List<? super T> list1, List<? extends T> list2) {}
}
```

则下面 6 段代码编译出错的是_____C E F_____。

A.

```
List<B> l1 = new ArrayList<> ();
List<B> l2 = new ArrayList<> ();
Test2_9.m (l1, l2);
```

B.

```
List<B> l3 = new ArrayList<> ();
List<D> l4 = new ArrayList<> ();
Test2_9.m (l3, l4);
```

C.

```
List<B> l5 = new ArrayList<> ();
List<A> l6 = new ArrayList<> ();
Test2_9.m (l5, l6);
```

D.

```
List<C> l7 = new ArrayList<> ();
List<D> l8 = new ArrayList<> ();
```

```
Test2_9.m (l7, l8);
E.
List<C> l7 = new ArrayList<> ();
List<D> l8 = new ArrayList<> ();
Test2_9. <B>m (l7, l8);
F.
List<D> l9 = new ArrayList<> ();
List<C> l10 = new ArrayList<> ();
Test2_9.m (l9, l10);
```

四、问答题

阅读下列程序，并填写表格

```
import java.util.*;
class A {}
class B extends A {}
class Test {
        public static void m1(List<? extends A> list) {}
        public static void m2(List<A> list) {}
        public static void m3(List<? super A> list) { }
        public static void main (String [] args) {
                List<A> listA = new ArrayList<A> ();
                List<B> listB = new ArrayList<B> ();
                List<Object> listO = new ArrayList<Object> ();

              // insert code here
      }
}
```

在上面代码插入点插入的代码	结果（从下面结果选项中选择）
m1(listA);	C
m2(listA);	C
m3(listA);	C
m1(listB);	C
m2(listB);	A
m3(listB);	A
m1(listO);	A
m2(listO);	A
m3(listO);	C
结果选项	
A. 编译出错	
B. 编译正确，运行出错	
C.编译正确，运行正确	

编程题 1:

```
class  ArrayIterator<T> implements Iterator<T>{
    private int pos = 0;
    private Object[] a = null;
    public ArrayIterator(Object[] array){
        a = array;
    }
    @Override
    public boolean hasNext() {
        return !(pos >= a.length);
    }
    @Override
    public T next() {
        if(hasNext()){
            T c = (T) a[pos];
            pos ++;
            return c;
        }
        else
            return null;
    }
}
public class Container<T> {
    private Object[] elements;
    private int elementsCount = 0;
    private int size = 0;
    public Container(int size){
        elements = new Object[size];
        this.size = size;
    }
    public boolean add(T e){
        if(elementsCount < size){
            elements[elementsCount ++] = e;
            return true;
        }
        else{
            return  false;
        }
    }
    public Iterator iterator(){
        return new ArrayIterator(elements);
    }
}
```

编程题 2:

```
public class TwoTuple<T1 extends Comparable<T1>, T2 extends Comparable<T2>> implements
Comparable<TwoTuple<T1,T2>>{
    private T1 first;
    private T2 second;
    public TwoTuple(T1 element1,T2 element2){
        first = element1;
        second = element2;
    }
    public void setFirst(T1 first) {this.first = first; }
    public void setSecond(T2 second) {this.second = second; }
    public T1 getFirst() {return first; }
    public T2 getSecond() {return second;}

    @Override public int compareTo(TwoTuple<T1,T2> o) {
        if(this.getFirst().compareTo(o.getFirst())==0){
            return this.getSecond().compareTo(o.getSecond());
        }else {
            return this.getFirst().compareTo(o.getFirst());
        }
    }

    @Override public boolean equals(Object obj) {
        if(obj instanceof TwoTuple){
            TwoTuple<T1,T2> o = (TwoTuple<T1,T2>) obj;
            return
o.getFirst().equals(this.getFirst())&&o.getSecond().equals(this.getSecond());
        }
        return false;
    }

    @Override public String toString() {
        return "("+first+","+second+")";
    }
}
```