一、填空

1. 语句Class clz = null；的含义是 clz是Class类型的空引用，没有指向任何Class类型的对象 。

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）第二个语句中clz3被泛型化为PolyGon，因此只能指向PolyGon类型信息，不能指向非PolyGon类型信息；第三个语句中clz4被泛型化为PolyGon类型以及他的子类，因此可以指向PolyGon类型以及它的子类，但不能指向父类 。。

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 前后参数泛型类型不统一。

使用泛型通配符?将错误的语句修改正确的方法是

```
ArrayList<? extends Object> lists = new ArrayList<String>();
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 变量类型不匹配，Integer类不能强制转换乘String类，

上面四段代码中编译不通过是③，原因是 无法推断参数类型，泛型不匹配，

这个例子说明泛型的作用是 类型检查以及不用强制类型转换。

二、单项选择题

1. 泛型参数代表的是（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 {}

B. interface Test2 {}

C. class Test3{ void test () {}}

D. class Test4{void 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（D）。无法将任何元素（`null`除外）放入`List<?>`中。

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

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

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

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

9. 给定下列代码：

```
class Shape {
}

class Circle extends Shape {
}

class Triangle extends Shape {
}

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 *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 void s3(T1 i, T1 j){}

使用泛型时，类型参数不允许为静态(`static`)。由于静态变量在对象之间共享，因此编译器无法确定要使用的类型

2. 下列语句编译时不出错的是AEGH_____。

```
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__ABC____。

A. List

B. List

C. List

D. List

4. 泛型参数应该写在的位置是BD_____。

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___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，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[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___CEF___。

A.

List **l1 = new ArrayList<> ();**

List l2 = new ArrayList<> ();

Test2_9.m (l1, l2);

B.

List l3 = new ArrayList<> ();

List l4 = new ArrayList<> ();

Test2_9.m (l3, l4);

C.

List l5 = new ArrayList<> ();

List l6 = new ArrayList<> ();

Test2_9.m (l5, l6);

D.

List l7 = new ArrayList<> ();

List l8 = new ArrayList<> ();

Test2_9.m (l7, l8);

E.

List l7 = new ArrayList<> ();

List l8 = new ArrayList<> ();

Test2_9. m (l7, l8);

F.

List l9 = new ArrayList<> ();

List l10 = new ArrayList<> ();

Test2_9.m (l9, l10);

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	