第二部分　Java 代码与编程部分

61、写一个 Singleton 出来

Singleton 模式主要作用是保证在 Java 应用程序中，一个类 Class 只有一个实例存在。

一般 Singleton 模式通常有几种种形式:

第一种形式: 定义一个类，它的构造函数为 private 的，它有一个 static 的 private 的该类变量，在类初始化时实例话，通过一个 public 的 getInstance 方法获取对它的引用,继而调用其中的方法。

```
public class Singleton {
private Singleton(){}
        //在自己内部定义自己一个实例，是不是很奇怪？
        //注意这是 private 只供内部调用
        private static Singleton instance = new Singleton();
        //这里提供了一个供外部访问本 class 的静态方法，可以直接访问
        public static Singleton getInstance() {
            return instance;
        }
    }
    第二种形式:
public class Singleton {
    private static Singleton instance = null;
    public static synchronized Singleton getInstance() {
    //这个方法比上面有所改进，不用每次都进行生成对象，只是第一次
    //使用时生成实例，提高了效率！
    if (instance==null)
        instance＝new Singleton();
return instance;        }
}
```

其他形式:

定义一个类，它的构造函数为 private 的，所有方法为 static 的。

一般认为第一种形式要更加安全些

62、继承时候类的执行顺序问题,一般都是选择题,问你将会打印出什么?

答:父类：

```
package test;
public class    FatherClass
{
    public FatherClass()
 {
   System.out.println("FatherClass Create");
 }
}
```

子类:

```
package test;
import test.FatherClass;
public class    ChildClass extends FatherClass
```

```
{
 public ChildClass()
 {
  System.out.println("ChildClass Create");
 }
 public static void main(String[] args)
 {
  FatherClass fc = new FatherClass();
  ChildClass cc = new ChildClass();
 }
}
```

输出结果：

```
C:\>java test.ChildClass
FatherClass Create
FatherClass Create
ChildClass Create
```

63、内部类的实现方式?

答：示例代码如下：

```
package test;
public class   OuterClass
{
 private class InterClass
 {
  public InterClass()
  {
   System.out.println("InterClass Create");
  }
 }
 public OuterClass()
 {
  InterClass ic = new InterClass();
  System.out.println("OuterClass Create");
 }
 public static void main(String[] args)
 {
  OuterClass oc = new OuterClass();
 }
}
```

输出结果:

```
C:\>java test/OuterClass
InterClass Create
OuterClass Create
```

再一个例题：

```
public class OuterClass {
   private double d1 = 1.0;
```

```
    //insert code here
}
```

You need to insert an inner class declaration at line 3. Which two inner class declarations are valid?(Choose two.)

A.
```
class InnerOne{
    public static double methoda() {return d1;}
}
```

B.
```
public class InnerOne{
    static double methoda() {return d1;}
}
```

C.
```
private class InnerOne{
    double methoda() {return d1;}
}
```

D.
```
static class InnerOne{
    protected double methoda() {return d1;}
}
```

E.
```
abstract class InnerOne{
    public abstract double methoda();
}
```

说明如下：

一.静态内部类可以有静态成员，而非静态内部类则不能有静态成员。 故 A、B 错

二.静态内部类的非静态成员可以访问外部类的静态变量，而不可访问外部类的非静态变量；return d1 出错。故 D 错

三.非静态内部类的非静态成员可以访问外部类的非静态变量。 故 C 正确

四.答案为 C、E

64、Java 的通信编程，编程题(或问答)，用 JAVA SOCKET 编程，读服务器几个字符，再写入本地显示？

答:Server 端程序:

```
package test;
import java.net.*;
import java.io.*;
public class Server
{
 private ServerSocket ss;
 private Socket socket;
 private BufferedReader in;
 private PrintWriter out;
 public Server()
 {
  try
  {
   ss=new ServerSocket(10000);
   while(true)
   {
    socket = ss.accept();
```

```
      String RemoteIP = socket.getInetAddress().getHostAddress();
      String RemotePort = ":"+socket.getLocalPort();
      System.out.println("A client come in!IP:"+RemoteIP+RemotePort);
      in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
      String line = in.readLine();
      System.out.println("Cleint send is :" + line);
      out = new PrintWriter(socket.getOutputStream(),true);
      out.println("Your Message Received!");
      out.close();
      in.close();
      socket.close();
    }
   }catch (IOException e)
   {
    out.println("wrong");
   }
  }
  public static void main(String[] args)
  {
   new Server();
  }
};
```

Client 端程序:

```
package test;
import java.io.*;
import java.net.*;
public class Client
{
  Socket socket;
  BufferedReader in;
  PrintWriter out;
  public Client()
  {
   try
   {
     System.out.println("Try to Connect to 127.0.0.1:10000");
     socket = new Socket("127.0.0.1",10000);
     System.out.println("The Server Connected!");
     System.out.println("Please enter some Character:");
     BufferedReader line = new BufferedReader(new
InputStreamReader(System.in));
     out = new PrintWriter(socket.getOutputStream(),true);
     out.println(line.readLine());
     in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
     System.out.println(in.readLine());
```

```
    out.close();
    in.close();
    socket.close();
   }catch(IOException e)
   {
    out.println("Wrong");
   }
  }
  public static void main(String[] args)
  {
   new Client();
  }
};
```

65、用 JAVA 实现一种排序，JAVA 类实现序列化的方法(二种)？ 如在 COLLECTION 框架中，实现比较要实现什么样的接口？

答:用插入法进行排序代码如下

```
package test;
import java.util.*;
class    InsertSort
{
  ArrayList al;
  public InsertSort(int num,int mod)
  {
   al = new ArrayList(num);
   Random rand = new Random();
   System.out.println("The ArrayList Sort Before:");
   for (int i=0;i<num ;i++ )
   {
    al.add(new Integer(Math.abs(rand.nextInt()) % mod + 1));
    System.out.println("al["+i+"]="+al.get(i));
   }
  }
  public void SortIt()
  {
   Integer tempInt;
   int MaxSize=1;
   for(int i=1;i<al.size();i++)
   {
          tempInt = (Integer)al.remove(i);
     if(tempInt.intValue()>=((Integer)al.get(MaxSize-1)).intValue())
     {
      al.add(MaxSize,tempInt);
      MaxSize++;
      System.out.println(al.toString());
     } else {
```

```
        for (int j=0;j<MaxSize ;j++ )
        {
          if
(((Integer)al.get(j)).intValue()>=tempInt.intValue())
          {
            al.add(j,tempInt);
            MaxSize++;
            System.out.println(al.toString());
            break;
          }
        }
       }
    }
    System.out.println("The ArrayList Sort After:");
    for(int i=0;i<al.size();i++)
    {
      System.out.println("al["+i+"]="+al.get(i));
    }
  }
  public static void main(String[] args)
  {
    InsertSort is = new InsertSort(10,100);
    is.SortIt();
  }
}
```