JAVA应用技术

复习课

知识点 I

- Java 常识
  - Java 代码编译执行的过程
    - 真实编译
    - 字节码解释执行
  - 跨平台的原因
    - JVM
    - 数据类型统一
  - Java 和 C++的比较
  - Java 内存模型
    - 对象都在堆里
    - 对象变量是指针
    - 垃圾回收机制
    - 数组下标检查
  - 单根结构：Object 类
  - main()
    - public static void main(String[] args)
    - 命令行参数
  - Java 关键字

- Java 基础：数据类型、对象和控制语句
  - 基本数据类型
  - 对象变量
    - 对象变量的意义
    - 对象变量的赋值
    - 对象变量做函数参数和返回值
    - 对象变量的比较
      - equals()
  - 字符串的连接
  - ?:运算符的结果类型问题
  - 带标号的 break 和 continue
- 类
  - this
    - 在成员函数内使用
    - 调用其他构造函数
  - 成员变量初始化
    - 定义初始化
    - 构造函数初始化
  - 静态成员
    - 静态成员的访问：通过“.”运算符
    - 静态成员变量和类对象的关系

知识点 II

- 数组
  - 数组的创建
  - 数组变量的赋值
  - 对象数组
  - for-each 循环
    - 对象数组 for-each 的特殊性
- 访问属性
  - import 的意义
  - package 和 CLASSPATH
  - 默认的访问属性：包内
  - protected：子类及包内
  - class 的访问属性
    - 默认的类仅限包内访问
    - public 类必须和源代码文件同名
- 继承和多态
  - 单继承
  - super 的作用
  - 继承和私有变量的关系
  - 和 C++的两个区别
    - 构造函数内实现了动态绑定
    - 没有名字隐藏
  - 默认动态绑定
  - final
    - final 变量
    - final 函数和类
- 特殊的类
  - 抽象
  - 接口
    - 接口作为数据类型
    - 接口可以多继承
    - 接口内的 default 函数
  - 内部类
    - 定义
    - 和外部类的关系
    - 匿名类的语法
  - 枚举类
    - 定义
    - 构造函数和成员函数
- 容器
  - 主要容器类型
    - List
    - Set
    - Map
  - 容器实现方式
    - ArrayList vs LinkedList
    - HashMap vs TreeMap
  - 遍历
    - Iterator
    - for-each
  - 范型的使用
  - 子类型范型和通配符

知识点 III

- 标准类库
  - String 类
    - 理解 String 是不可写的对象
    - 常用函数
    - 在 switch-case 中使用
  - StringBuffer 类
  - Random 类
- 异常
  - throw-try-catch 机制
    - throw
    - catch 的匹配方式
    - 万能 catch
    - Throwable 接口的方法
  - finally
  - 函数对抛出异常的声明 throws
    - 编译时检查
    - 与构造函数的关系
    - 与函数覆盖的关系

- IO
  - stream：只处理 byte
    - 文件流的使用
    - 流的基本函数
  - Reader/Writer 和 stream 的关系
    - 通过桥建立两者的关系
    - 如何做汉字编码转换
  - DataInput/OutputStream
    - 理解二进制流
  - 对象串行化

- GUI
  - 部件、容器、布局管理器的关系
  - JFrame 类的使用
    - add()
    - pack()
    - setDefaultCloseOperation()
  - Graphics 类的使用
    - 理解 paint()函数
  - 常见布局管理器的效用
  - 菜单类族的使用
  - Swing 的消息机制
    - 消息机制
    - Listener、Event 类
    - add/removeListener 函数
    - 理解以线程方式通知
  - 常见部件（略）
  - JTable 与 MVC 模式

知识点 IV

- 线程
  - 创建线程：Runnable、Thread
  - 线程控制：start()、sleep()、yield()
  - 线程同步：synchronized
  - 线程的 wait()和 notify()机制
  - 通过管道的线程间通信
- RTTI
  - Class 类
    - getClass()
    - .class
    - isIntance()
    - 从 Class 类对象中获得父类、接口和函数的方法
  - instanceof 运算符
- socket 通信
  - TCP 的 Socket 和 ServerSocket
  - UDP 的通信方式
  - 构建 socket 服务的设计模式

- JDBC
  - JDBC 如何连接和查询
  - 事务处理
  - preparedStatement
- 函数式编程
  - Lambda 表达式
  - 函数式接口
- 流式计算
  - 容器的 stream 接口
  - 常用的高阶函数
    - 过滤
    - 映射
    - 聚合

样题

判断题

`char` of Java is 8-bit. (1分)	F
A Java class can extend from multiple base classes. (1分)	F
Member variables are to get default init values when the object is to be created. (1分)	T
`protected` member can be visited by extended class only. (1分)	F
InputStream and OutputStream read and write 8-bit data. (1分)	T
Swing container is used to organize other GUI components in. But other containers can not be put in a container. (1分)	F
To access a method of a class, an object of that class must be created first. (1分)	F
When an object is de-serialized, its constructor does not run. (1分)	T

样题

没有implements Serializable

```
public class ParentDeserializationTest {

public static void main(String[] args){
    try {
        System.out.println("Creating...");
        Child c = new Child(1);
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        ObjectOutputStream oos = new ObjectOutputStream(baos);
        c.field = 10;
        System.out.println("Serializing...");
        oos.writeObject(c);
        oos.flush();
        baos.flush();
        oos.close();
        baos.close();
        ByteArrayInputStream bais = new ByteArrayInputStream(baos.toByteArray());
        ObjectInputStream ois = new ObjectInputStream(bais);
        System.out.println("Deserializing...");
        Child c1 = (Child)ois.readObject();
        System.out.println("c1.i="+c1.getI());
        System.out.println("c1.field="+c1.getField());
    } catch (IOException ex){
        ex.printStackTrace();
    } catch (ClassNotFoundException ex){
        ex.printStackTrace();
    }
}
```

```
public static class Parent {
    protected int field;
    protected Parent(){
        field = 5;
        System.out.println("Parent::Constructor");
    }
    public int getField() {
        return field;
    }
}

public static class Child extends Parent implements Serializable{
    protected int i;
    public Child(int i){
        this.i = i;
        System.out.println("Child::Constructor");
    }
    public int getI() {
        return i;
    }
}
```

Output:

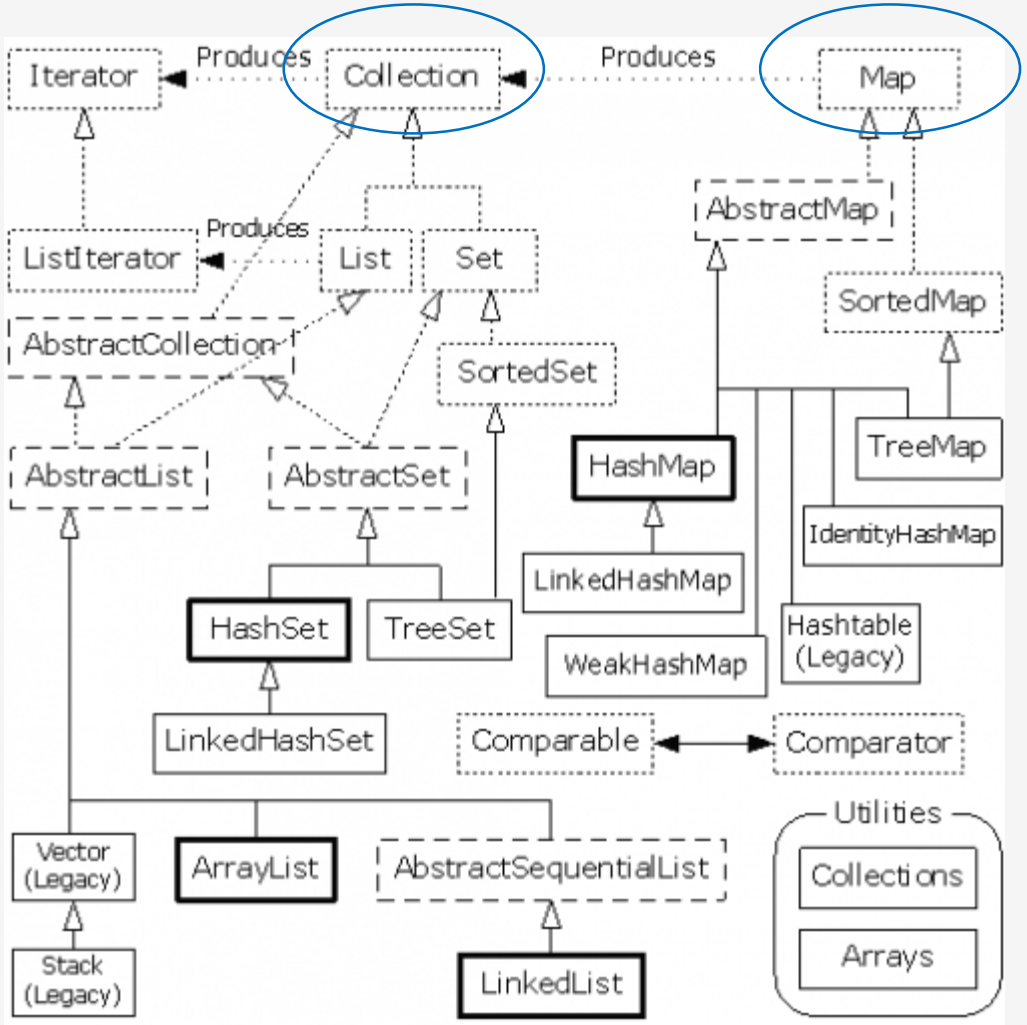
```
Creating...
Parent::Constructor
Child::Constructor
Serializing...
Deserializing...
Parent::Constructor
c1.i=1
c1.field=5
```

样题

单选题

About Java containers, which statement below is NOT correct? (2分)

- ○ A. `List` holds the elements in a particular sequence
- ○ B. `Set` cannot have any duplicate elements
- ○ C. `Map` has group of key-value object pairs
- ◉ D. `Iterator` can deal with `List`, `Set` and `Map`



HashMap

```
public Set<K> keySet() {
    Set<K> ks = keySet;
    if (ks == null) {
        ks = new KeySet();
        keySet = ks;
    }
    return ks;
}
```

```
public Collection<V> values() {
    Collection<V> vs = values;
    if (vs == null) {
        vs = new Values();
        values = vs;
    }
    return vs;
}
```

```
public Set<Map.Entry<K,V>> entrySet() {
    Set<Map.Entry<K,V>> es;
    return (es = entrySet) == null ? (entrySet = new EntrySet()) : es;
}
```

样题

For swing event handling mechanism, which one below is NOT correct? (2分)

- ○ A. Event source like JButton is able to have more than one ActionListener objects registered
- ○ B. When an event occurs, the source object notices all the registered listeners
- ○ C. A registered listener is able to be de-registered from a source object dynamically
- ◉ D. One listener can not be registered at more than one source object

For code below:

```
ArrayList<Integer> a = new ArrayList<Integer>();
ArrayList<Double> b = new ArrayList<Double>();
```

Which statement below is **NOT** correct? (2分)

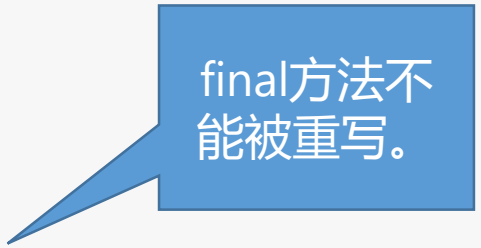
- ○ A. a.getClass().equals(b.getClass()) is true
- ○ B. a.getClass() == b.getClass() is true
- ○ C. a instanceof ArrayList is true
- ◉ D. a.getClass() == b.getClass() is false

样题

Given code below:

```
class Base{
        public final void method() {
                System.out.println("Base.method");
        }
}

public final class Fin extends Base{
        public void method() {
                System.out.println("Fin.method");
        }
        public static void main(String argv[]){
                Base b = new Fin();
                b.method();
        }
}
```



While one below is correct? (2分)

- ○ A. It does not compile because of method() in Fin is not defined final as its base one
- ○ B. It does not compile because Fin can not be final
- ◉ C. It does not compile because of method() in Base final so no function can override it in derived classes
- ○ D. It compiles and prints Fin.method

样题

For InputStream.read(), the read() with no parameters, which statement below is correct? (2分)

- (●) A. read() returns int, because it has to return EOF to indicate the end of the file
- ( ) B. read() returns byte, because it reads a byte from the stream
- ( ) C. read() returns char, because it reads a char from the stream
- ( ) D. read() returns int, as the number of bytes it just read

The value returned is a byte as an int type.

java.io.InputStream	
+read(): int	Reads the next byte of data from the input stream. The value byte is returned as an int value in the range 0 to 255. If no byte is available because the end of the stream has been reached, the value –1 is returned.
+read(b: byte[]): int	Reads up to b.length bytes into array b from the input stream and returns the actual number of bytes read. Returns -1 at the end of the stream.
+read(b: byte[], off: int, len: int): int	Reads bytes from the input stream and stores into b[off], b[off+1], …, b[off+len-1]. The actual number of bytes read is returned. Returns -1 at the end of the stream.
+available(): int	Returns the number of bytes that can be read from the input stream.
+close(): void	Closes this input stream and releases any system resources associated with the stream.
+skip(n: long): long	Skips over and discards n bytes of data from this input stream. The actual number of bytes skipped is returned.
+markSupported(): boolean	Tests if this input stream supports the mark and reset methods.
+mark(readlimit: int): void	Marks the current position in this input stream.
+reset(): void	Repositions this stream to the position at the time the mark method was last called on this input stream.

样题

Given the following code:

```
public class Test {
        String s;
        static class Inner {
                void testMethod() {
                        s = "Hello world.";
                }
        }
        public static void main(String[] argv) {
                Inner i = new Inner();
                i.testMethod();
                System.out.println(s);
        }
}
```

- Since a static inner class has no connection to an object of the outer class, within an inner class method
  - Instance variables of the outer class **cannot be** referenced
  - Nonstatic methods of the outer class **cannot be** invoked

which one below is correct?(2分)

- ○ A. It compiles and prints out Hello world.
- ◉ B. It does not compile because String s in class Test is not static.
- ○ C. It does not compile because Inner can not used in the way in main()
- ○ D. It compiles and exception raises at running indicates that s has not been initiated.

样题

Given the following class definition which of the following can be legally placed after the comment line //Here ?(2分)

```
class Base{
        public Base(int i){}
}

public class MyOver extends Base{
        public static void main(String arg[]){
                MyOver m = new MyOver(10);
        }
        MyOver(int i){
                super(i);
        }
        MyOver(String s, int i){
                this(i);
                //Here
        }
}
```

- ○ A. `MyOver m = new MyOver();`
- ○ B. `super();`
- ○ C. `this("Hello",10);`
- ◉ D. `Base b = new Base(10);`

样题

Why might you define a method as native? (2分)

- ◉ A. To get to access hardware that Java does not know about
- ○ B. To define a new data type such as an unsigned integer
- ○ C. To write optimized code for performance in a language such as C/C++
- ○ D. To overcome the limitation of the private scope of a method

For exception, which statement below is **NOT** correct? (2分)

- ○ A. It is possible to have a try block with out any catch clause but a finally clause
- ○ B. It is possible to have a try block inside another try block
- ○ C. It is possible to have a try block along with its catch clauses inside a catch clause
- ◉ D. To re-throw the exception object in a catch clause, simple put a throw statement without the name of the object.

Which of the following will output -3.0 (2分)

- ○ A. System.out.println(Math.floor(-3.7));
- ○ B. System.out.println(Math.round(-3.7));
- ◉ C. System.out.println(Math.ceil(-3.7));
- ○ D. System.out.println(Math.min(-3.7));

样题

What must be done when throwing an integer as an exception? (2分)

- (●) A. Integers cannot be thrown.
- ( ) B. Declare integers as Throwable.
- ( ) C. Import the exception class.
- ( ) D. Encapsulate the integer handler

What best describes the appearance of an application with the following code?

```
public class FlowAp extends Frame{
public static void main(String argv[]){
    FlowAp fa=new FlowAp();
    fa.setSize(400,300);
    fa.setVisible(true);

}

FlowAp(){
        add(new Button("One"));
        add(new Button("Two"));
        add(new Button("Three"));
        add(new Button("Four"));
    }
}
```

(2分)

- ( ) A. A Frame with buttons marked One to Four placed on each edge.
- ( ) B. A Frame with buttons marked One to four running from the top to bottom
- (●) C. A Frame with one large button marked Four in the Centre
- ( ) D. An Error at run time indicating you have not set a LayoutManager

样题

What will happen when you attempt to compile and run the following code?

```
public class Bground extends Thread{
        public static void main(String argv[]){
                Bground b = new Bground();
                b.run();
        }
        public void start(){
                for (int i = 0; i <10; i++){
                        System.out.println("Value of i = " + i);
                }
        }
}
```

(2分)

- ○ A. A compile time error indicating that no run method is defined for the Thread class
- ○ B. A run time error indicating that no run method is defined for the Thread class
- ○ C. Clean compile and at run time the values 0 to 9 are printed out
- ◉ D. Clean compile but no output at runtime

样题

Suppose there is no file Hello.txt in the current directory. Run the program: (2分)

```
import java.io.*;
public class ABC {
	public static void main(String argv[]) throws Exception {
		ABC m=new ABC();
		System.out.println(m.ff());
	}

	public int ff() {
		try {
			FileInputStream dis=new FileInputStream("Hello.txt");
		} catch (FileNotFoundException fne) {
			System.out.print("No such file found, ");
			throw fne;
		} finally {
			System.out.print("Doing finally, ");
		}
		return 0;
	}
}
```

- ○ A. No such file found,
- ○ B. No such file found ,0
- ◉ C. No such file found, Doing finally,
- ○ D. No such file found, Doing finally, 0

样题

About layout manager in AWT and Swing, which one below is correct? (2分)

- ○ A. `FlowLayout` is the default layout manager of `Frame`.
- ◉ B. `GridLayout` divides the whole space into even pieces.
- ○ C. It is not possible to specify coordinates of component regardless the effect of any layout managers.
- ○ D. Every place in a `BorderLayout` has to be fill with a component, or it will leave blank.

Which statement below is NOT correct? (2分)

- ○ A. A thread is an instance of Thread class.
- ○ B. A thread runs the run() method of the Runnable object.
- ◉ C. A new born thread can run immediately when start() is called.
- ○ D. Thread can access data of the Runnable object.

Given code below:

```
List<String> ls = new ArrayList<String>();
List<Object> lo = ls;
lo.add(new Object());
String s = ls.get(0);
```

Which statement below is correct? (2分)

- ◉ A. It does not compile
- ○ B. It compiles but exception raises at line 2
- ○ C. It compiles but exception raises at line 3
- ○ D. It compiles but exception raises at line 4

样题

程序输出题

请写出以下程序运行结果：

```
public class X {
    public static void main(String [] args) {
        try {
            badMethod();
            System.out.print("A");
        } catch (RuntimeException ex) {
            System.out.print("B");
        } catch (Exception ex1) {
            System.out.print("C");
        } finally {
            System.out.print("D");
        }
        System.out.print("E");
    }
    public static void badMethod() {
        throw new RuntimeException();
}}
```

BDE (2分)

请写出以下程序运行结果：

```
class Test {
    public static void main(String[] args) {
        Integer a = new Integer(3);
        Integer b = 3;
        int c = 3;
        System.out.println(a == b);
        System.out.println(a == c);
}}
```

false (2分)

true (2分)

请写出以下程序运行结果：

```
public class Test {
   public static void main(String[]args){
      House house1 = new Test().new House(1,100);
      House house2 = (House)house1.clone();
      System.out.println(house1==house2);
      System.out.println(house1.equals(house2));
      System.out.println(house1.whenBuilt==house2.whenBuilt);
      System.out.println(house1.whenBuilt.equals(house2.whenBuilt));
   }
   public class House implements Cloneable, Comparable<House> {
      private int id;
      private int area;
      private java.util.Date whenBuilt;
      public House(int id, int area) {
         this.id = id;
         this.area = area;
         whenBuilt = new java.util.Date();
      }
      @Override
      public Object clone() {
         try {
            House houseClone = (House)super.clone();
            houseClone.whenBuilt = (java.util.Date) (whenBuilt.clone());
            return houseClone;
         } catch (CloneNotSupportedException ex) {
            return null;
      }}
      @Override
      public int compareTo(House o) {
         if (area > o.area)
            return 1;
         else if (area < o.area)
            return -1;
         else
            return 0;
}}}
```

false (2分)

false (2分)

false (2分)

true (2分)

equals没有重写

样题

请写出以下程序运行结果：

```
enum EnumTry {
   MON, TUE, WED, THU, FRI;
   public static void main(String[] args)  {
      for (EnumTry e : EnumTry.values()) {
         System.out.println(
            e +":"+ e.toString() +":"+ e.ordinal() +":"+ e.name());
}}}
```

MON:MON:0:MON (2分)

TUE:TUE:1:TUE (2分)

WED:WED:2:WED (2分)

THU:THU:3:THU (2分)

FRI:FRI:4:FRI (2分)

样题

给出以下代码：

```
public class Test {
        public int t=4;
        public static void main(String[] args) {
                new Test().NumberPlay();
        }
        public void NumberPlay() {
                int t=2;
                t = t+5;
                this.t = this.t-2;
                t = t-this.t;
                System.out.println(t+this.t+"ok");
        }
}
```

程序运行后输出结果为：7ok (2分)

请写出以下程序运行结果：

```
class Main  {
        public static void main(String[] args) {
                String s1 = "Zhejiang University";
                String s2 = s1.substring(0, 7);
                s2.toUpperCase();
                System.out.println(s2+s1.substring(8));
        }
}
```

Zhejian University (2分)

样题

请写出以下程序运行结果：

```
public class Test {
        public static void main(String[] args) throws Exception{
            String str = "hello";
            Method m = str.getClass().getMethod("toUpperCase");
            System.out.println(m.invoke(str));
            System.out.println(str);
}}
```

HELLO (2分)

hello (2分)

```
public enum Main {
  PLUS   { int eval(int x, int y) { return x + y; } },
  MINUS  { int eval(int x, int y) { return x - y; } },
  TIMES  { int eval(int x, int y) { return x * y; } },
  DIVIDE { int eval(int x, int y) { return x / y; } };
  abstract int eval(int x, int y);
  public static void main(String args[]) {
                int x = 4;
        int y = 2;
        for (Main op : Main.values())
            System.out.printf("%d %s %d = %d%n", x, op, y, op.eval(x, y));
    }
}
```

程序运行结果为（一行一空）：

(2分)

(2分)

(2分)

(2分)