Programs and JDK-Documentation

Examination in Object Oriented Programming S 2016

University of Applied Science Ravensburg-Weingarten
Prof. Dr. M. Zeller

Please do not write on these sheets.

1 Program Exception

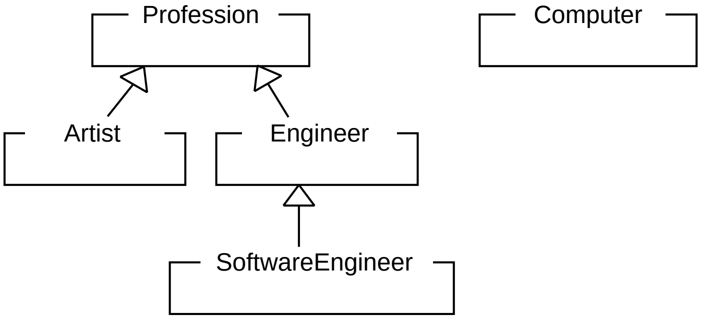
```
public class WriteException extends Exception{
}

public class ReadException extends WriteException{
}

public class PrintException extends ReadException{
}

public void foo(int num) throws Exception {
        System.out.println("num: " + num);
        if (num == 0) {
            throw new WriteException();
        }
        if (num == 1) {
            int res = 10 / (num - 1);
        }
        if (num == 2) {
            throw new PrintException();
        }
        if (num == 3) {
            throw new ReadException();
        }
        if (num == 4) {
            int[] numbers = new int[2];
            numbers[num] = num;
        }
    }

public void bar() {
        for (int i = 0; i < 5; i++) {
            try {
                foo(i);
            } catch

        ⋮

        }
    }
}
```

2 Program Constructors

All classes belong to the same package.

```java
public class Profession {
    String name = "cook-general";
    int level;
    Profession() {
        System.out.println("creating a profession");
        level = 3;
    }
    public void work() {
        System.out.println("working as: " + name + ", " + level);
    }
}

public class Engineer extends Profession {
    Engineer() {
        System.out.println("Creating an engineer, " + level);
        name = "engineer";
    }
    void designSystem() {
        work();
        System.out.println("designing a system: " + name);
    }
}

public class SoftwareEngineer extends Engineer {
    Computer theComputer;

    SoftwareEngineer() {
        System.out.println("creating a programmer ");
    }
    SoftwareEngineer(int aLevel) {
        theComputer = new Computer();
        name = "sw designer";
        System.out.println("creating a sw-engineer ");
        level = aLevel;
    }
    public void coding() {
        System.out.println("coding, typing source code");
    }
    public void work() {
        System.out.println(name + ": working, designing sw, " + level);
    }
}

public class Artist extends Profession {
    public void work() {
        perform();
        System.out.println(", enjoying applause, ");
    }
    public void perform(){
        System.out.print("performing, " + level);
    }
                :
                :
}
```

```java
public class Computer {
    Computer(){
        System.out.println("creating a computer");
    }
}
```



```java
public class ProfessionTest {

    public static void main(String[] args) {
        ProfessionTest pt = new ProfessionTest();
        pt.testProfessions();
        return;
    }
    public void testProfessions() {
        System.out.println(" 1 ——————————————");
        Profession aProfession = new Profession();
        System.out.println(" 2 ——————————————");
        aProfession.work();
        System.out.println(" 3 ——————————————");
        Engineer anEngineer = new Engineer();
        System.out.println(" 4 ——————————————");
        anEngineer.work();
        System.out.println(" 5 ——————————————");
        anEngineer.designSystem();
        System.out.println(" 6 ——————————————");
        SoftwareEngineer aSwEngineer = new SoftwareEngineer(7);
        System.out.println(" 7 ——————————————");
        aSwEngineer.work();
        System.out.println(" 8 ——————————————");
        aSwEngineer.designSystem();
        System.out.println(" 9 ——————————————");
        aSwEngineer.coding();
        System.out.println("10 ——————————————");
        Profession aPerson = new Artist();
        System.out.println("11 ——————————————");
        aPerson.work();
        // aPerson.getPhone();
        // Artist anArtist = new Artist();
        // anArtist.setPhone("0751 501 . . .");
        // String aPhone = anArtist.phone;
        // int aLevel = anArtist.level;
    }
}
```

3 Program Tree

```java
public class Tree {
    Branch center;

    public static void main(String[] args) {
        Tree aTree = new Tree();
        aTree.makeTree();
        aTree.printTree(aTree.center);
    }

    public void makeTree() {
        Branch branch_1 = new Branch(31);
        Branch branch_2 = new Branch(22);
        Branch branch_3 = new Branch(43);
        Branch branch_4 = new Branch(14);
        Branch branch_5 = new Branch(35);

        center = branch_3;
        branch_3.left = branch_2;
        branch_3.right = branch_1;
        center.left.right = branch_5;
        branch_1.right = branch_4;
    }

    void printTree(Branch aBranch){

          ⋮

    }
}

public class Branch {
    int value;
    Branch left;
    Branch right;

    Branch(int theValue){
        value = theValue;
    }

    void print(){
        System.out.println("Branch: " + value);
    }
}
```

4 Program Collections and IO

```
public class FileIoTest {

    String truckFileName = "truckDataFile.dat";

    . . . . . . . . .. . . . . . . . .  truckList;
    TruckIO truckIoMgr;

    FileIoTest() {
        truckList  = . . . . . . . .. . . . . . . .
        truckIoMgr = new TruckIO();
    }

    public static void main(String[] args) {
        FileIoTest fiot = new FileIoTest();
        fiot.createRandomTrucks(5);
        fiot.printTrucks();
        fiot.saveTrucks();
        fiot.truckList.remove(1);
        fiot.truckList.remove(1);
        fiot.printTrucks();
        fiot.readTrucks();
        fiot.printTrucks();
    }

    void saveTrucks() {
        try (DataOutputStream dos = truckIoMgr.getDataOutputStream(truckFileName)) {
            dos.writeInt(truckList.size());
            for (Truck aTruck : truckList) {
                truckIoMgr.saveTruckToStream(aTruck, dos);
            }
        } catch (IOException ioex) {
            ioex.printStackTrace();
        }
     }

     void readTrucks() {
        try (DataInputStream dis = truckIoMgr.getDataInputStream(truckFileName)) {
            int truckCount = dis.readInt();
            for (int i = 0; i < truckCount; i++) {
                Truck aTruck = truckIoMgr.readTruckFromStream(dis);
                truckList.add(aTruck);
            }
        } catch (IOException ioex) {
            ioex.printStackTrace();
        }
     }
}
```

```
public class Truck{
    String identifier;
    double capacity;
    byte stateCode;

    void print() {
        System.out.print("Truck: "  . . . . . .
        . . . . . . . . . . . . . . . . . . . .
        . . . . . . . . . . . . . . . . . . . .
    }
}

public class TruckIO {

    DataOutputStream getDataOutputStream(String fileName) throws IOException {
        FileOutputStream fos =  . . . . . . . . . . . . . .
        . . . . . . . . . . . . . . . . . . . . . . . . . .
        DataOutputStream dos =  . . . . . . . . . . . . . .
        return dos;
    }

    DataInputStream getDataInputStream(String fileName) throws IOException {
        FileInputStream fis =  . . . . . . . . . . . . . .
        . . . . . . . . . . . . . . . . . . . . . . . . . .
        DataInputStream dis = . . . . . . . . . . . . . .
        return dis;
    }

    void saveTruckToStream(Truck aTruck, DataOutputStream dos) throws IOException {
        . . . . . . . . . . . . . .
        . . . . . . . . . . . . . .
        . . . . . . . . . . . . . .
    }

    Truck readTruckFromStream(DataInputStream dis) throws IOException {
        Truck aTruck = new Truck();
        . . . . . . . . . . . . . .
        . . . . . . . . . . . . . .
        . . . . . . . . . . . . . .
        return aTruck;
    }
}
```

5 Program Class Design

```
public class Eagle{
    public void fly() {
        System.out.println("an eagle flies");
    }
}

public class Condor {
    public void fly() {
        System.out.println("a condor flies");
    }
}

public class Flamingo{
   public void fly() {
        System.out.println("a flamingo flies");
    }
}

public class LetFlyExample {

    int objCount = 5;
    Condor[]   condors   = new Condor[objCount];
    Eagle[]    eagles    = new Eagle[objCount];
    Flamingo[] flamingos = new Flamingo[objCount];

    public static void main(String[] args) {
        LetFlyExample lfe = new LetFlyExample();
        lfe.createAndLetFly();

    }

    public void createAndLetFly() {
        for (int i = 0; i < objCount; i++) {
            condors[i] = new Condor();
        }
        for (int i = 0; i < objCount; i++) {
            eagles[i] = new Eagle();
        }
        for (int i = 0; i < objCount; i++) {
            flamingos[i] = new Flamingo();
        }

        for (int i = 0; i < objCount; i++) {
            condors[i].fly();
            eagles[i].fly();
            flamingos[i].fly();
        }
```

6 FileOutputStream Summary

A file output stream is an output stream for writing data to a File ...

6.1 Constructors (Selection)

FileOutputStream(File file)
Creates a file output stream to write to the file represented by the specified File object.

FileOutputStream(String name)
Creates a file output stream to write to the file with the specified name.

7 BufferdOutputStream Summary

The class implements a buffered output stream. By setting up such an output stream, an application can write bytes to the underlying output stream without necessarily causing a call to the underlying system for each byte written.

7.1 Constructors (Selection)

BufferedOutputStream(OutputStream out)
Creates a new buffered output stream to write data to the specified underlying output stream.

BufferedOutputStream(OutputStream out, int size)
Creates a new buffered output stream to write data to the specified underlying output stream with the specified buffer size.

8 FileInputStream

A FileInputStream obtains input bytes from a file in a file system ...

8.1 Constructors (Selection)

FileInputStream(String name) Creates a FileInputStream by opening a connection to an actual file, the file named by the path name name in the file system.

9 BufferedInputStream

A BufferedInputStream adds functionality to another input stream – namely, the ability to buffer the input ...

9.1 Constructors (Selection)

BufferedInputStream(InputStream in) Creates a BufferedInputStream and saves its argument, the input stream in, for later use.

10 DataOutputStream Summary

```
public class DataOutputStream
```

10.1 Constructor

DataOutputStream(OutputStream out)
Creates a DataOutputStream that uses the specified underlying OutputStream.

10.2 Methods (Selection)

void	flush() Flushes this data output stream.
int	size() Returns the current value of the counter written, the number of bytes written to this data output stream so far.
void	write(byte[] b, int off, int len) Writes len bytes from the specified byte array starting at offset off to the underlying output stream.
void	write(int b) Writes the specified byte (the low eight bits of the argument b) to the underlying output stream.
void	writeBoolean(boolean v) Writes a boolean to the underlying output stream as a 1-byte value.
void	writeByte(int v) Writes out a byte to the underlying output stream as a 1-byte value.
void	writeBytes(String s) Writes out the string to the underlying output stream as a sequence of bytes.
void	writeChar(int v) Writes a char to the underlying output stream as a 2-byte value, high byte first.
void	writeChars(String s) Writes a string to the underlying output stream as a sequence of characters.
void	writeDouble(double v) Converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the underlying output stream as an 8-byte quantity, high byte first.
void	writeFloat(float v) Converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the underlying output stream as a 4-byte quantity, high byte first.
void	writeInt(int v) Writes an int to the underlying output stream as four bytes, high byte first.
void	writeLong(long v) Writes a long to the underlying output stream as eight bytes, high byte first.
void	writeShort(int v) Writes a short to the underlying output stream as two bytes, high byte first.
void	writeUTF(String str) Writes a string to the underlying output stream using modified UTF-8 encoding in a machine-independent manner.

11 DataInputStream Summary

```
public class DataInputStream
```

11.1 Constructor

DataInputStream(InputStream in)
Creates a DataInputStream that uses the specified underlying InputStream.

11.2 Methods (Selection)

int	read(byte[] b) Reads some number of bytes from the contained input stream and stores them into the buffer array b.
int	read(byte[] b, int off, int len) Reads up to len bytes of data from the contained input stream into an array of bytes.
boolean	readBoolean() Reads one input byte and returns true if that byte is nonzero, false if that byte is zero.
byte	readByte() Reads and returns one input byte.
char	readChar() Reads two input bytes and returns a char value.
double	readDouble() Reads eight input bytes and returns a double value.
float	readFloat() Reads four input bytes and returns a float value.
void	readFully(byte[] b) Reads some bytes from an input stream and stores them into the buffer array b.
void	readFully(byte[] b, int off, int len) Reads len bytes from an input stream.
int	readInt() Reads four input bytes and returns an int value.
String	readLine() Reads the next line of text from the input stream.
long	readLong() Reads eight input bytes and returns a long value.
short	readShort() Reads two input bytes and returns a short value.
int	readUnsignedByte() Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.
int	readUnsignedShort() Reads two input bytes and returns an int value in the range 0 through 65535.
String	readUTF() See the general contract of the readUTF method of DataInput.
static String	readUTF(DataInput in) Reads from the stream in a representation of a Unicode character string encoded in modified UTF-8 format; this string of characters is then returned as a String.

12 ArrayList Summary

```
public class ArrayList<E>
```

12.1 Constructors

ArrayList()
Constructs an empty list with an initial capacity of ten.

ArrayList(Collection<? extends E> c)
Constructs a list containing the elements of the specified collection, in the order they are returned by the collection's iterator.

ArrayList(int initialCapacity)
Constructs an empty list with the specified initial capacity.

12.2 Methods (Selection)

boolean	add(E e) Appends the specified element to the end of this list.
void	add(int index, E element) Inserts the specified element at the specified position in this list.
E	get(int index) Returns the element at the specified position in this list.
E	remove(int index) Removes the element at the specified position in this list.
int	size() Returns the number of elements in this list.