Examination in Object Oriented Programming WS 2014

Programs and JDK-Documentation

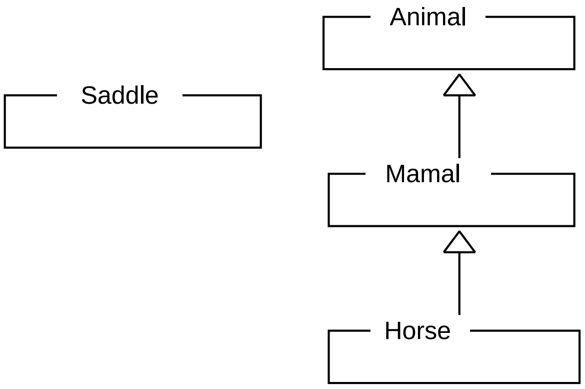
University of Applied Science Ravensburg-Weingarten
Prof. Dr. M. Zeller

Note: Please do not write any answers to these sheets.

1 Program Constructors

```java
package biology;
public class Animal {
    public Animal() {
        System.out.println("  in Animal()");
    }
    public void print() {
        System.out.println("This is an Animal");
        return;
    }
}

package biology;
public class Mamal extends Animal {
    protected String name = "unnamed";
    private int age;

    public Mamal() {
        System.out.println("  in Mamal()");
        age = 11;
    }
    public void print() {
        System.out.println("  Name: " + name);
    }
    public void printAll() {
        print();
        System.out.println("  Age: " + age);
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
}

package biology;
public class Horse extends Mamal {
    private Saddle theSaddle = new Saddle();
    Horse() {
        System.out.println("  in Horse()");
        name = "Morningstar";
    }
    public void print() {
        System.out.println(" Horse:");
        super.print();
        theSaddle.print();
    }
    public void changeSaddle(String newColor, int newSize) {
        theSaddle.color = newColor;
        theSaddle.size = newSize;
    }
}
```

```
package biology;
public class Saddle {
    String color = "brown";
    int size = 2;

    public Saddle() {
        System.out.println("  in Saddle()");
    }
    public void print() {
        System.out.print("  Saddle color: " + color);
        System.out.print(", size:" + size);
    }
    public void setSize(int size) {
        this.size = size;
    }
}
```



The `main()`-Method:

```
public static void main(String[] args) {
    Horse myHorse = new Horse();
    Mamal myMamal = myHorse;
    System.out.println("———— 1 ——————");
    myHorse.print();
    System.out.println();
    System.out.println("———— 2 ——————");
    myHorse.changeSaddle("dark brown", 3);
    myMamal.printAll();
    System.out.println("———— 3 ——————");
}
```

```
package admin;
import biology.Horse;
public class HorseAdmin {

    public void rideHorse(String name_) {
        Horse aHorse = new Horse();
        aHorse.name = name_;
        if (aHorse.age < 20){
            System.out.println(" riding OK");
        } else{
            System.out.println(" horse is too old for riding");
        }
        aHorse.print();
    }
}
```

2 Program Railroad

```java
class Waggon {
    Waggon follower;
    int weight = 12345;

    Waggon(int weight_){
        weight = weight_;
    }
}

public class Locomotive {
    int weight;

    Locomotive(int weight_){
        weight = weight_;
    }
}

public class Train {
    int trainID;
    Locomotive theLocomotive;
    Waggon theWaggons;

    Train(int id) {
        trainID = id;
    }
    void addLocomotive(Locomotive newLocomotive){
            :
    }
    void addWaggon(Waggon newWaggon){
            :
    }
    int getWeight(){
            :
    }
}

public static void main(String[] args) {
    Train aTrain = new Train(42);
    int trainWeight = aTrain.getWeight();
    System.out.println(" (1) Weight: " + trainWeight);
    Locomotive aLocomotive = new Locomotive(21);
    aTrain.addLocomotive(aLocomotive);
    Waggon aWaggon = new Waggon(12);
    aTrain.addWaggon(aWaggon);
    aWaggon = new Waggon(22);
    aTrain.addWaggon(aWaggon);
    trainWeight = aTrain.getWeight();
    System.out.println(" (2) Weight: " + trainWeight);
    aWaggon = new Waggon(11);
    aTrain.addWaggon(aWaggon);
    trainWeight = aTrain.getWeight();
    System.out.println(" (3) Weight: " + trainWeight);
}
```

3 Program Exception

```
package exceptiontest;

public class CenterException extends Exception{
}

public class LeftException extends CenterException {
    int exNum;
    LeftException(int number) {
        exNum = number;
    }
}

public class RightException extends CenterException {
}

public class ExceptionProgram {
    int numbers[] = {1, 2, 3, 4, 5};

    public ExceptionProgram() {
    }

    public static void main(String[] args) {
        ExceptionProgram exProgObj = new ExceptionProgram();
        exProgObj.foo(0);
        exProgObj.foo(1);
        exProgObj.foo(2);
        exProgObj.foo(3);
        exProgObj.foo(4);
        exProgObj.foo(5);
        return;
    }
    void foo(int num) {
        try {
            int result = bar(num);
            System.out.println("in foo, result: " + result);
        }
```

— catch clauses go here ... —

— ...but please write your answers on the exam sheet – not on this sheet —

```
        return;
    }

    int bar(int num) throws Exception {
        int result = numbers[num] / (2 - num);
        if (num == 0) {
            throw new RightException();
        }
        if (num == 2) {
            throw new CenterException();
        }
        if (num == 3) {
            throw new LeftException(num);
        }
        if (num == 4) {
            throw new CenterException();
        }
        return result;
    }
}
```

4 Program Collection and IO

```java
public class Person {

    static String[] firstNameArray = {"Anna", "Bob", "Claire", "Don", "Elisa"};
    static String[] lastNameArray = {"Kent", "Lewis", "Mills", "North", "Owen"};

    String firstName;
    String lastName;
    short yearOfBirth;

    Person(String firstName_, String lastName_, short yearOfBirth_) {
        firstName = firstName_;
        lastName = lastName_;
        yearOfBirth = yearOfBirth_;
    }

    static Person createRandomPerson() {
        int index = getRandomInt(firstNameArray.length - 1);
        String fName = firstNameArray[index];
        index = getRandomInt(lastNameArray.length - 1);
        String lName = lastNameArray[index];
        short yob = (short) (1915 + getRandomInt(100));
        Person aPerson = new Person(fName, lName, yob);
        return aPerson;
    }

    // generate a random integer value, 0 <= value <= max
    static int getRandomInt(int max) {
        return (int) Math.round(Math.random() * max);
    }

    void print() {
        System.out.print(firstName + " ");
        System.out.print(lastName + ", born in ");
        System.out.println(yearOfBirth);
    }
}
```

```
public class PersonAdmin {
    ArrayList<Person> persList;

    public PersonAdmin() {
        persList = new ArrayList<Person>(40);
    }

public static void main(String[] args) {
    PersonAdmin pao = new PersonAdmin();
    pao.initializePersonList(3);
    pao.printPersons();
    Person aPerson = new Person("Finn", "Pony", (short) 2015);
    pao.persList.add(1, aPerson);
    pao.persList.remove(3);
    pao.printPersons();
    // − − − 1 − − −
    pao.savePersToFile("persFile.dat");
    pao.persList.clear();
    pao.readPersFromFile("persFile.dat");
    pao.printPersonsToFile("persPrintFile.txt");
    aPerson = Person.createRandomPerson();
    pao.persList.add(10, aPerson);
    return;
}

    public void initializePersonList(int persCnt) {
        for (int i = 0; i < persCnt; i++) {
            Person aPerson = Person.createRandomPerson();
            persList.add(aPerson);
        }
        return;
    }

    public void printPersons() {
        System.out.println(" —— List of persons —— ");
            :
            :

        return;
    }

BufferedOutputStream getBufferedOutputStream(String fileName)
                    throws IOException {
                        :
                        :

    BufferedOutputStream bos = . . . .
    return bos;
}
public void savePersToFile(String dataFileName) {
    DataOutputStream dos = null;
    try {
        BufferedOutputStream bos = getBufferedOutputStream(dataFileName);
        dos = . . . . . . . . . . . . . . ;     // create output stream
        . . . . . . . . . . . . . . . . . . . . // save number of objects
        for (int i = 0; i < . . . . . . . . . . . . ; i++) {
            Person aPerson =  . . . . . . . . . . // pick one object
            dos . . . . . . . . . . . . . . . . . // save yearOfBirth);
            dos . . . . . . . . . . . . . . . . . // save firstName);
            dos . . . . . . . . . . . . . . . . . // save lastName);
        }
    } catch (IOException ex) {
        ex.printStackTrace();
    }
    try {
        dos.close();
    } catch (IOException ex) {
        ex.printStackTrace();
    }
    return;
}

    BufferedInputStream getBufferedInputStream(String fileName)
                    throws IOException {
                        :
                        :
        BufferedInputStream bis = . . . .
        return bis;
    }

public void readPersFromFile(String dataFileName) {
    DataInputStream dis = null;
    try {
        BufferedInputStream bis = getBufferedInputStream(dataFileName);
        dis = new DataInputStream(bis);
        int persCnt = . . . . . . . . . . . .    // read number of data sets
        for (int i = 0; i < persCnt; i++) {
            short yob = . .  . . . . . . . . . .    // read short value
            String fName = . . . . . . . . . . .    // read string
            String lName = . . . . . . . . . . .    // read string
            Person aPerson = . . . . . . . . . .    // create new person
            persList . . . . . . . . . . . . . .    // add to persList
        }
    } catch (IOException ex) {
        ex.printStackTrace();
    }
    try {
        dis.close();
    } catch (IOException ex) {
        ex.printStackTrace();
    }
    return;
}
```

5 ArrayList Summary

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

5.1 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.

5.2 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.

6 FileOutputStream Summary

```
public class FileOutputStream
```

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

```
public class BufferedOutputStream
```

7.1 Constructors

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 DataOutputStream Summary

```
public class DataOutputStream
```

8.1 Constructor

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

8.2 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.

9 DataInputStream Summary

```
public class DataInputStream
```

9.1 Constructor

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

9.2 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.