Programs and JDK-Documentation

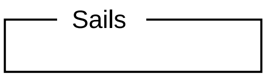
Examination in Object Oriented Programming S 2015

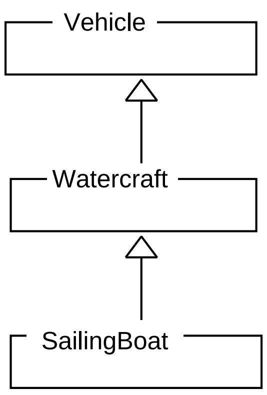
University of Applied Science Ravensburg-Weingarten
Prof. Dr. M. Zeller

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

1 Program Constructors

```java
package vehicles;
class Vehicle {
    int idNum = 1;
    Vehicle() {
        System.out.println("Creating Vehicle: " + idNum);
    }
    void move(){
        System.out.println("Moving Vehicle: " + idNum);
    }
}

class Watercraft extends Vehicle {
    Watercraft() {
        System.out.println("Creating Watercraft: " + idNum);
    }
}

class SailingBoat extends Watercraft {
    Sails theSails = new Sails();
    SailingBoat() {
        idNum = 20;
        System.out.println("Creating SailingBoat: " + idNum);
    }
    void sail(){
        System.out.print("Sailing SailingBoat: " + idNum);
        System.out.println(", sails:  " + theSails.area);
    }
    void move(){
        System.out.println("Moving SailingBoat: " + idNum);
    }
}

class Sails {
    float area = 30;
    Sails() {
        System.out.println("Creating Sails: ");
    }
    public float getArea(){
        return area;
    }
}

public class Inheritance {

    public static void main(String[] args) {
        SailingBoat mayFlower = new SailingBoat();
        mayFlower.sail();
        Watercraft bounty      = new SailingBoat();
        bounty.move();

    }
}
```





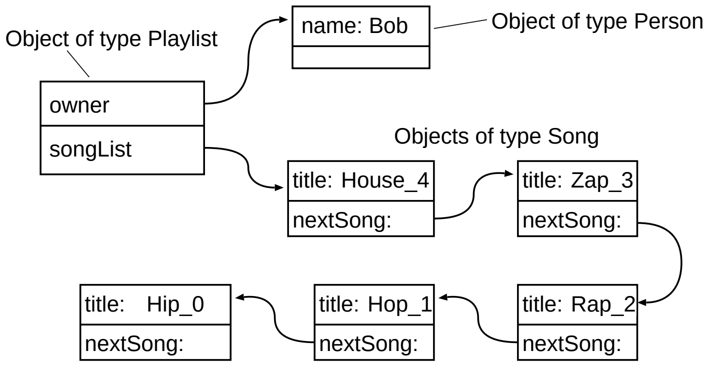
2 Program Exception

```
class NorthException extends Exception{
}

class SouthException extends Exception{
}

class NorthWestException extends NorthException{
}

public class ExceptionTest {

    public static void main(String[] args) {
        try{
            int result = foo(1);
        }catch(

        ⋮

        }
    }
}
```

3 Program Playlist

```
class Person {
    String name;
    Person(String theName) {
        name = theName;
    }
    public void print() {
        System.out.println(name + " plays for us ");
    }
}

class Song {
    Song nextSong;
    String title;
    Song(String theTitle) {
        title = theTitle;
    }
    public void play() {
        System.out.println("playing: " + title);
    }
}

public class PlayList {
    Person owner;
    Song songList;
    PlayList(){
    }
    PlayList(Person theOwner) {
        owner = theOwner;
    }
    public void addSong(Song aSong) {
            . . .
    }
    public Song removeSong() {
            . . .
    }
    public void play() {
            . . .
    }
    public void play(int index) {
            . . .
    }
}
```

```java
public static void main(String[] args) {
        Person aPerson = new Person("Bob");
        PlayList aPlayList = new PlayList(aPerson);
        for(int i = 0; i < 5; i++){
            Song aSong = createSong(i);
            aPlayList.addSong(aSong);
        }
        aPlayList.play();
        aPlayList.play(3);
        Song aSong = aPlayList.removeSong();
        System.out.println("removed: " + aSong.title);
        aSong =  aPlayList.removeSong();
        System.out.println("removed: " + aSong.title);
        aPlayList.play();
        aPlayList = new PlayList();
        aPlayList.play();
        return;
    }
```



The data structure after the program has completed the for-loop in the method `main()`.
The output of the program is:

```
 --------- playing playlist --------
Bob plays for us
playing: House_4
playing: Zap_3
playing: Rap_2
playing: Hop_1
playing: Hip_0

playing song 3 --------
playing: Hop_1
removed: House_4
removed: Zap_3

 --------- playing playlist --------
Bob plays for us
playing: Rap_2
playing: Hop_1
playing: Hip_0

--------- playing playlist --------
```

4 Program Collection and IO

```
class Customer {
    int customerId;
    String name;
    float volume;

    Customer(int id, String theName, float theVolume) {
        customerId = id;
        name = theName;
        volume = theVolume;
    }

    void print() {
        System.out.println();
        System.out.print("ID: " + customerId + ", Name: " + name);
        System.out.println(", Revenue: " + volume);
    }
}

class CustomerIO {
    void writeCustomerFile(int custCount, String fileName) {
        try (DataOutputStream dos = getDataOutputStream(fileName)) {
            for (int i = 0; i < custCount; i++) {
                Customer aCustomer = createRandomCustomer();
                saveCustomer(aCustomer, dos);
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

   void saveCustomer(Customer aCustomer, DataOutputStream dos) throws IOException {
       dos . . . . . . . . . . . . . . . . . . . . .
       dos . . . . . . . . . . . . . . . . . . . . .
       dos . . . . . . . . . . . . . . . . . . . . .
    }

    DataOutputStream getDataOutputStream(String fileName) {
         DataOutputStream dos = null;
         try {
             FileOutputStream fos = . . . . . . . . . . . . .
             BufferedOutputStream bos = . . . . . . . . . . . . .
             dos = . . . . . . . . . . . . .
         } catch (IOException ioex) {
             ioex.printStackTrace();
         }
         if ( . . . . . . . . . . . . . . ) {
             System.out.println("could not get OutputStream: " + fileName);
             System.exit(-1);
        }
        return dos;
    }
    DataInputStream getDataInputStream(String fileName) {
        DataInputStream dis = null;
        try {
            FileInputStream fis = . . . . . . . . . . . . . . . .
            BufferedInputStream bis = . . . . . . . . . . . . . . . .
            dis = . . . . . . . . . . . . . . .

        } catch (IOException ioex) {
            ioex.printStackTrace();
        }
        if ( . . . . . . . . . . . . . . . . ) {
            System.out.println("could not get InputStream: " + fileName);
            System.exit(-1);
        }
        return dis;
    }

    Customer readCustomer(DataInputStream dis) {
        Customer aCustomer = null;
        try {
            int theId          = . . . . . . . . . . . . . . .
            String theName     = . . . . . . . . . . . . . . .
            float theVolume    = . . . . . . . . . . . . . . .
            aCustomer          = . . . . . . . . . . . . . . .
            System.out.println("id: " + theId);
        } catch (EOFException ex) {
            System.out.println("done reading file: ");
        }
        catch (IOException ioex) {
            ioex.printStackTrace();
        }
        return aCustomer;
    }

    void readCustomerFile(String fileName, . . . . . . . . . . . customerMap) {
        try (DataInputStream dis = getDataInputStream(fileName)) {
            Customer aCustomer;
            while ((aCustomer = readCustomer(dis)) != null) {
                customerMap. . . . . . . . . . . . . . . . . . . ;
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return;
    }

    Customer createRandomCustomer() {
                 . . .
    }
}
```

```
public class PersonDB {

    . . . . . . . . . . . . . . . . . . . . . . customerMap;
    CustomerIO custIO;
    String fileName;

    PersonDB() {
        custIO = new CustomerIO();
        fileName = "customer.dat";
    }

    public static void main(String[] args) {
        PersonDB pdb = new PersonDB();
        pdb.testCustomerIO();
    }

    void testCustomerIO() {
        custIO.writeCustomerFile(5, fileName);
        customerMap = . . . . . . . . . . . . . . . . . . . . . . .;
        custIO.readCustomerFile(fileName, customerMap);
        System.out.println("Number of customers: " + customerMap.size());
        selectCustomer();
        Customer aCustomer = custIO.createRandomCustomer();
        aCustomer.print();
        customerMap.put(12345, aCustomer);
        aCustomer = custIO.createRandomCustomer();
        aCustomer.print();
        customerMap.put(12345, aCustomer);
        aCustomer = customerMap.get(12345);
        aCustomer.print();
        aCustomer = customerMap.get(1111);
    }
```

5 FileOutputStream Summary

```
public class FileOutputStream
```

5.1 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.

6 BufferdOutputStream Summary

```
public class BufferedOutputStream
```

6.1 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.

7 DataOutputStream Summary

```
public class DataOutputStream
```

7.1 Constructor

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

7.2 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.

8 DataInputStream Summary

```
public class DataInputStream
```

8.1 Constructor

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

8.2 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.

9 HashMap Summary

9.1 Constructors (Selection)

HashMap()
Constructs an empty HashMap with the default initial capacity (16) and the default load factor (0.75).

HashMap(int initialCapacity)
Constructs an empty HashMap with the specified initial capacity and the default load factor (0.75).

9.2 Methods (Selection)

V	get(Object key) Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
V	put(K key, V value) Associates the specified value with the specified key in this map. If the map previously contained a mapping for the key, the old value is replaced.
V	remove(Object key) Removes the mapping for the specified key from this map if present.
int	size() Returns the number of key-value mappings in this map.
Collection<V>	values() Returns a Collection view of the values contained in this map.