Programs and JDK-Documentation

Examination in Object Oriented Programming W 2015

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

Please do not write on these sheets.

1 Program Exception

```
1  public class GameException extends Exception{
2  }
3
4  public class MoveException extends GameException{
5  }
6
7  public class TimeOutException extends MoveException{
8  }
9
10 public void foo(int num) throws Exception {
11         System.out.println("num: " + num);
12         if (num == 0) {
13             throw new GameException();
14         }
15         if (num == 1) {
16             int res = 10 / (num - 1);
17         }
18         if (num == 2) {
19             throw new TimeOutException();
20         }
21         if (num == 3) {
22             throw new MoveException();
23         }
24         if (num == 4) {
25             int [] numbers = new int[2];
26             numbers[num] = num;
27         }
28     }
29
30 public void bar() {
31         for (int i = 0; i < 5; i++) {
32             try {
33                 foo(i);
34             } catch

            .
            .
            .

35         }
36     }
37 }
```

2 Program Constructors

```
1  public class Game {
2      String name = "Game of life";
3      int difficulty;

5      Game() {
6          System.out.println("Creating a game");
7          difficulty = 3;
8      }

10     public void start() {
11         System.out.println("starting: " + name + ", " + difficulty);
12     }
13 }

15 public class BoardGame extends Game {
16     BoardGame() {
17         System.out.println("Creating a board game, " + difficulty);
18         name = "board game";
19     }

21     void placeBoard() {
22         System.out.println("placing board for game: " + name);
23     }
24 }

26 public class Chess extends BoardGame {
27     Chess() {
28         name = "Chess";
29         System.out.println("creating a chess game ");
30         difficulty = 5;
31     }

33     public void placeChessmen() {
34         System.out.println("placing chessmen");
35     }

37     public void start() {
38         System.out.println(name + ": white's move");
39     }
40 }

42 public class CardGame extends Game {
43     public void start() {
44         shuffle();
45         System.out.println("starting card game, " + difficulty);
46     }

48     public void shuffle(){
49         System.out.print("shuffle cards, ");
50     }
51 }
```

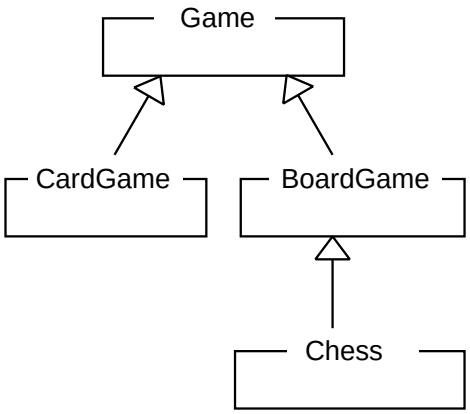

Figure 1: The class hierarchy of the program

```
52 public void testGames() {
53     Game aGame = new Game();
54     BoardGame aBoardGame = new BoardGame();
55     Chess aChessGame = new Chess();
56     System.out.println("----------------------");
57     aGame.start();
58     aBoardGame.start();
59     aChessGame.start();
60     System.out.println("----------------------");

62     aBoardGame.placeBoard();
63     aChessGame.placeBoard();
64     aChessGame.placeChessmen();

66     Game theGame = new CardGame();
67     theGame.start();
68 }
```

3 Program Tree

```
1  public class Node {
2      int key;
3      Node greater;
4      Node smaller;

6      Node(int theKey){
7          key = theKey;
8      }
9  }

11 public class Tree {
12     Node root;
13     public static void main(String[] args) {
14         Tree aTree = new Tree();
15         aTree.makeTree();
16     }

18     public void makeTree() {

20         Node node_1 = new Node(20);
21         Node node_2 = new Node(35);
22         Node node_3 = new Node(30);
23         Node node_4 = new Node(15);
24         Node node_5 = new Node(25);

26         root = node_1;
27         node_1.greater = node_3;
28         node_1.smaller = node_4;
29         root.greater.smaller = node_5;
30         node_3.greater = node_2;

               ⋮

31     }
32 }
```

4 Program Collection and IO

```
1  public class FileIoTest {
2      String toyFileName = "toyPrintFile.dat";

4      . . . . . toyList;

6      FileIoTest() {
7          toyList = . . . . . .
8      }

10     public static void main(String[] args) {
11         FileIoTest fiot = new FileIoTest();
12         fiot.makeToys();
13     }

15     void makeToys() {
16         createRandomToys(5);
17         printToys();
18         ToyIO toyio = new ToyIO();
19         try (PrintStream ps = toyio.getPrintStream(toyFileName)) {
20             for (Toy aToy : toyList) {
21                 toyio.printToyToStream(aToy, ps);
22             }
23         } catch (IOException ioex) {
24             ioex.printStackTrace();
25         }
26         toyList.remove(2);
27         toyList.remove(2);
28         printToys();
29         try (Scanner fileScanner = toyio.getFileScanner(toyFileName)) {
30             while (fileScanner.hasNext()) {
31                 Toy aToy = toyio.readToy(fileScanner);
32                 toyList.add(aToy);
33             }
34         } catch (IOException ioex) {
35             ioex.printStackTrace();
36         }
37         printToys();
38     }

40     void printToys() {
41         System.out.println("Toylist: ");
42         for (Toy aToy : toyList) {
43             aToy.print();
44         }
45         System.out.println("----------------------");
46     }

48     void createRandomToys(int toyCount) {
49         . . .
50     }
51 }
```

```
52 public class Toy {
53     String name;
54     int colorCode;
55     float weight;

57     void print() {
58         System.out.print("Toy: " . . . .
59         . . . . . .
60         . . . . . .
61     }
62 }

63 public class ToyIO {
64     PrintStream getPrintStream(String fileName) throws IOException {
65         FileOutputStream fos = . . . . . .
66         . . . . . .
67         PrintStream ps = . . . . . .
68         return ps;
69     }

71     void printToyToStream(Toy aToy, PrintStream ps) {
72         . . . . . .
73         . . . . . .
74         . . . . . .
75     }

77     Scanner getFileScanner(String fileName) throws IOException {
78         FileInputStream fis = . . . . . .
79          . . . . . .
80          . . . . . .
81         return fs;
82     }

84     Toy readToy(Scanner aScanner) {
85         Toy aToy = new Toy();
86         . . . . . .
87         . . . . . .
88         . . . . . .
89         . . . . . .
90         . . . . . .
91         . . . . . .
92         return aToy;
93     }
94 }
```

5 Program Class Design

```
1  public class Circle {
2      Point center;
3      double radius;
4      public void draw() {
5          System.out.println("drawing a circle");
6      }
7  }

9  public class Line {
10     Point start;
11     Point end;
12     public void draw() {
13         System.out.println("drawing a line");
14     }
15 }

17 public class Rectangle {
18     Point leftUpper;
19     Point rightLower;
20     public void draw(){
21         System.out.println("drawing a rectangle");
22     }
23 }

25 public class GeometryTest {
26     int objCount = 5;
27     Line[]          lines    = new Line[objCount];
28     Circle[]        circles  = new Circle[objCount];
29     Rectangle[] rectangles = new Rectangle[objCount];

31     public static void main(String[] args) {
32         GeometryTest gt = new GeometryTest();
33         gt.testGeometry();

35     }

37     public void testGeometry() {
           ⋮        // populate arrays: lines, circles and rectangles
38         for (int i = 0; i < objCount; i++) {
39             lines[i].draw();
40             circles[i].draw();
41             rectangles[i].draw();
42         }
           ⋮
43         }
44     }
45 }
```

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 Scanner

A simple text scanner which can parse primitive types and strings using regular expressions.

A Scanner breaks its input into tokens using a delimiter pattern, which by default matches whitespace. The resulting tokens may then be converted into values of different types using the various next methods.

10.1 Constructors (Selection)

Scanner(File source) Constructs a new Scanner that produces values scanned from the specified file.

Scanner(InputStream source) Constructs a new Scanner that produces values scanned from the specified input stream.

10.2 Methods (Selection)

boolean	hasNext() Returns true if this scanner has another token in its input.
String	next() Finds and returns the next complete token from this scanner.
boolean	nextBoolean() Scans the next token of the input into a boolean value and returns that value.
byte	nextByte() Scans the next token of the input as a byte.
double	nextDouble() Scans the next token of the input as a double.
float	nextFloat() Scans the next token of the input as a float.
int	nextInt() Scans the next token of the input as an int.

11 PrintStream Summary

```
public class PrintStream
```

11.1 Constructors (Selection)

PrintStream(File file)
Creates a new print stream, without automatic line flushing, with the specified file.

PrintStream(OutputStream out)
Creates a new print stream.

11.2 Methods (Selection)

void	print(boolean b) Prints a boolean value.
void	print(char c) Prints a character.
void	print(char[] s) Prints an array of characters.
void	print(double d) Prints a double-precision floating-point number.
void	print(float f) Prints a floating-point number.
void	print(int i) Prints an integer.
void	print(long l) Prints a long integer.
void	print(Object obj) Prints an object.
void	print(String s) Prints a string.
void	println() Terminates the current line by writing the line separator 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.