Chapter 5

Some useful classes

Lesson page 5-1. Numerical wrapper classes

Activity 5-1-1 Wrapper-class Integer

Question 1. False. A new Integer can be created, but its contents cannot be changed.

Question 2. d = new Integer(d.intValue() - 5);

Activity 5-1-2 Instance methods of class Integer

Question 3. Use its toString method, e.g. (new Integer(5)).toString()

Activity 5-1-3 Static constants and methods of class Integer

Question 4. System.out.println(Integer.MIN_VALUE);

Question 5. Integer.toString(270) or "+ 270 or
(new Integer(270)).toString()

Question 6. Integer.parseInt("456") or
(new Integer("456")).intValue() .

Lesson page 5-2. Wrapper classes Boolean and Character

Lesson page 5-3. Strings

Question 1. A literal is a Java denotion for a value of a type.

Question 2. Single-quote char.: \’ double-quote char.: \" backslash: \\ newline: \n
Question 3. The value is 9.

Question 4. The length is 7. The expression is "abcdcefg".length() .
Activity 5-3-1 String literals

Question 5. Here's the table:

<table>
<thead>
<tr>
<th>Output</th>
<th>Java String</th>
</tr>
</thead>
<tbody>
<tr>
<td>What?!?</td>
<td>&quot;What?!?&quot;</td>
</tr>
<tr>
<td>Who's this?</td>
<td>&quot;Who's this?&quot; OR &quot;Who's this?&quot;</td>
</tr>
<tr>
<td>When are they arriving?</td>
<td>&quot;When are they arriving?&quot;</td>
</tr>
<tr>
<td>Sage said &quot;Ah!&quot; and ate.</td>
<td>&quot;Sage said &quot;Ah!&quot; and ate.&quot;</td>
</tr>
</tbody>
</table>

Activity 5-3-2 Variables of type String

Question 6. String name = "Petra";

Question 7. ""Isn't there more?" they chorused."

Activity 5-3-3 Operation catenation

Question 8. System.out.println(name + "'s job is " + job + ")."]

Activity 5-3-4 Operation catenation (continued)

Question 9. When performing catenation, when using System.out.print, and when using System.out.println.

Question 10. Average: 1 depth - 4.52 speed - 0.2

Activity 5-3-5 Referencing the characters of a String

Question 11. The result is: C

Question 12. System.out.print("My ");

String s77 = "da ";
String s2 = "is ";
String s25 = "n" + s77.charAt(1) + "me ";
System.out.print(s25);
System.out.print(s2);
String s5 = "Leo" + s25.substring(0,2);
System.out.print(s5);
String s18 = "r" + s77.substring(0,1) + s5.charAt(2);
System.out.print(s18 + s2.charAt(2));
System.out.print(s77);
System.out.print(s23);
System.out.println(".");

Activity 5-3-6 Equality of Strings

Question 13. It will print true. In Java, two String literals (or, more generally, constant expressions like "a" + "bc") that are equal (using method equals) will share a unique instance of class String.
Chapter 5. Some useful classes 51

Question 14. As in the previous question, it will print true. However, the following will print false, because the concatenation is not a constant expression.

```java
String s = "Sam";
String t = "Sa";
System.out.println(s == t + "m");
```

Question 15. The body of method equals is:

```java
return numerator == f.numerator &&
        denominator == f.denominator;
```

Question 16. Here is method equals:

```java
public boolean equals(Book b) {
    return (title.equals(b.title) &&
            numPages == b.numPages &&
            price == b.price);
}
```

Question 17. This exercise is most easily done using loops. However, it can be done without loops using methods in class String. Read first method lastVowel, below, then method firstVowel.

```java
public class Test {
    // Using a String literal, swap the first and last vowels
    // and print the literal and the result.
    public static void main(String []args) {
        String s = "put your string here";
        int first = firstVowel(s);
        int last = lastVowel(s);
        String t = s;

        // if t has more than one vowel,
        // swap the first and last
        if (first != last) {
            t = t.substring(0, first) +
                t.charAt(last) + t.substring(first+1,last) +
                t.charAt(first) + t.substring(last+1);
        }
        System.out.println(s + " " + t);
    }

    // = index of first vowel in s, -1 if no vowels
    public static int firstVowel(String s) {
        s = s.toLowerCase();
        int i = s.indexOf('a');
        if (i == -1) i = s.length();
    }
```
```java
int ie = s.indexOf('e');
if (ie != -1) i = Math.min(i, ie);

int ii = s.indexOf('i');
if (ii != -1) i = Math.min(i, ii);

int io = s.indexOf('o');
if (io != -1) i = Math.min(i, io);

int iu = s.indexOf('u');
if (iu != -1) i = Math.min(i, iu);

if (i == s.length()) i = -1;
return i;
}

// = index of last vowel in s, -1 if no vowels
public static int lastVowel(String s) {
    s = s.toLowerCase();
    int i = Math.max(s.lastIndexOf('a'),
                     s.lastIndexOf('e'));
    i = Math.max(i, s.lastIndexOf('i'));
    i = Math.max(i, s.lastIndexOf('o'));
    i = Math.max(i, s.lastIndexOf('u'));
    return i;
}
```

Lesson page 5-4. Class StringBuffer

**Question 1.** Changeable.

**Question 2.** An object is immutable if it cannot be changed.

**Question 3.** Below is method main:

```java
public static void main(String args[]) {
    StringBuffer word = new StringBuffer("envelope");
    if (word.length() > 5) {
        word.reverse();
        word.insert(0, "i");
    }
    System.out.println(word);
}
```
Lesson page 5-5. Class Vector

Question 1. Capacity: the number of elements that are allocated.

Activity 5-5-1 Adding objects to a Vector

Question 2. False. The capacity increases automatically.
Question 3. True.
Question 4. The statement v.addElement(3); is illegal; the argument cannot be of a primitive type.
Question 5. Missing the import statement.
Question 6. Missing the import statement.

Activity 5-5-2 Referencing and changing objects in a Vector

Question 7. x = (Integer) v.elementAt(4);
Question 8. Every instance of class Object contains a toString method, and every class is a subclass of Object. Method elementAt returns an Object, and the print statement automatically calls toString. If the real class of the object has its own toString, that is the method which is executed.
Question 9. Here’s the sequence:

```java
int sum = ((Integer)v.elementAt(0)).intValue();
sum = sum + ((Integer)v.elementAt(1)).intValue();
v.setElementAt(new Integer(sum), 2);
```

Activity 5-5-3 Other Vector methods

Question 10. False. The size will be 4 but the capacity will be 6.
Question 11. Here is the method; its specification is in the Companion.

```java
public static String vectorWarp(Vector v) {
    int sz = v.size();
    if(sz==0) {
        System.out.println("The vector is empty.");
    }
    else if(sz%2 == 0) {
        System.out.println("The vector is even.");
        Object o = v.lastElement();
        v.removeElementAt(sz-1);
        v.insertElementAt(o, 0);
    }
    else {
        System.out.println("This was an odd vector.");
        v.removeElementAt((sz-1)/2);
    }
}```
Chapter 5. Some useful classes

```java
}
String vStr= v.toString();
v.removeAllElements();
return vStr;
```

Lesson page 5-6. Class Date

Question 1. True.

```java
Date today= new Date();
System.out.println("Today is " + today);
```

Lesson page 5-7. Reading from the keyboard and files

Question 1. A stream is a sequence of values that are processed from beginning to end.

Question 2. An input stream is a stream whose values are read.

Question 3. An output stream is a stream whose values are written. Thus, an output stream usually starts out empty and increases in size as values are written to it.

Activity 5-7-1 Linking to the keyboard

Question 4. Standard input is usually the keyboard.

Question 5. System.in is the standard input.

Question 6. The statement is:
```java
InputStreamReader isr= new InputStreamReader(System.in);
```

Question 7. The statement is:
```java
BufferedReader br= new BufferedReader(new InputStreamReader(System.in));
```

Activity 5-7-2 Reading a line at a time from the keyboard

Question 8. String line= br.readLine();

Activity 5-7-3 Handling an IO exception

Question 9. import java.io.*;

Question 10. throws IOException

Question 11. ONLY ONE! Your program will be hopelessly confused if more than one part of it is trying to read from the keyboard. If you need
a BufferedReader that reads from the keyboard in more than one method, pass it in as a parameter to that method.

Activity 5-7-4 Reading numbers

Question 12. The argument " 77 " has contains whitespace; it shouldn’t.

Question 13. Here’s method main:

```java
public static void main(String[] args)
    throws IOException {
    BufferedReader br=
        new BufferedReader
            (new InputStreamReader(System.in));
    int sum= Integer.parseInt(br.readLine().trim());
    sum= sum + Integer.parseInt(br.readLine().trim());
    System.out.println(sum);
}
```

Activity 5-7-5 Reading from a file

Question 14. One was linked to the keyboard, the other to a file.

Question 15. The program obtained the name of the file from the user.

Question 16. When there are no more lines, readLine returns null.

Activity 5-7-6 Using a dialog box

Question 17. Here are the statements:

- JFileChooser jd = new JFileChooser();
- jd.setDialogTitle("Choose input file");
- jd.showOpenDialog(null);
- jd.getSelectedFile() (which can be used to create a new FileReader, which is used to create a new BufferedReader.)

Question 18. Here is class main and a method getReader:

```java
public static void main(String[] args) throws IOException {
    BufferedReader f1= getReader();
    BufferedReader f2= getReader();
    String s1= f1.readLine();
    String s2= f2.readLine();
    if (s1.equals(s2))
        { System.out.println("same"); }
    else {System.out.println("different"); }
}
```
public static BufferedReader getReader() throws IOException {
    JFileChooser jd1 = new JFileChooser();
    jd1.setDialogTitle("Choose first input file");
    jd1.showOpenDialog(null);
    return new BufferedReader(
        new FileReader(jd1.getSelectedFile()));
}

Lesson page 5-8. Writing to the Java console and files

Activity 5-8-1 Writing a file

Question 1. PrintStream ps = new PrintStream(
        new FileOutputStream("f.txt"));

Question 2. Here's the answer:

// Read a line from a file that the user chooses and write
// that line to another file that the user chooses.
public static void questionTwo() throws Exception {

    // Get input and output file names from user and
    // create links to the files
    BufferedReader br = new BufferedReader(
        new InputStreamReader(System.in));
    System.out.print("Read from file name: ");
    BufferedReader fromFile = new BufferedReader(
        new FileReader(br.readLine()));
    System.out.print("Write to file name: ");
    PrintStream toFile = new PrintStream(
        new FileOutputStream(br.readLine()));

    // Read first line and write it to the file.
    toFile.println(fromFile.readLine());
}

Question 3. Here's the answer:

// Read a line from a file that the user chooses and
// append that line to another file that the user chooses.
public static void questionThree() throws Exception {

    // Get input and output file names from user and create
    // links to the files
BufferedReader br = new BufferedReader(
    new InputStreamReader(System.in));
System.out.print("Read from file name: ");
BufferedReader fromFile = new BufferedReader(
    new FileReader(br.readLine()));
System.out.print("Write to file name: ");
PrintStream toFile = new PrintStream(
    new FileOutputStream(br.readLine(), true));

// Read first line and write it to the file.
toFile.println(fromFile.readLine());

Question 4. Here's the answer:

// Read a line from a file that the user chooses, ask
// whether to append or to overwrite, then append
// or overwrite the line to another file chosen
// by the user.
public static void questionFour() throws Exception {

    // Get input file name from user link to the file.
    BufferedReader br = new BufferedReader(
        new InputStreamReader(System.in));
    System.out.print("Read from file name: ");
    BufferedReader fromFile = new BufferedReader(
        new FileReader(br.readLine()));

    // Get output file name from user, and ask if they
    // want to overwrite or append, and link accordingly.
    System.out.print("Write to file name: ");
    String outFile = br.readLine();
    System.out.print(      
        "Do you wish to overwrite contents of " +  
        outFile + "? y/n: ");
    PrintStream toFile; // the output file stream
    if ("y".equals(  
        ((br.readLine()).trim()).toLowerCase()) ) {
        System.out.println("Overwriting file " +  
            outFile + ".");
        toFile = new PrintStream(  
            new FileOutputStream(outFile));
    } else {  
        System.out.println("Appending to file " +  
            outFile + ".");
        toFile = new PrintStream(  
            new FileOutputStream(outFile, true));
    }
}
new FileOutputStream(outFile, true));
}

// Read first line and write it to the file.
toFile.println(fromFile.readLine());
}