Chap09Yourname.java

char and Strings

```
/*
 * Chap09 char and String
 */

import java.io.PrintStream;
import java.util.*;
//We may import java.util.Scanner, java.util.Arrays and java.util.Random

public class Chap09 {
    private static Scanner in;
    public static void main (String[] args) {
        PrintStream out = System.out;
        String fruit, name, reName, upperName;
        char letter;
        int n, length, index, diff, hour, minute;
        in = new Scanner(System.in);
```

9.1 Characters, type char.

```
out.println("9.1 Characters");
//Java provides 8 primitive data types for representing integers, real number,
//characters, and Boolean values:
//byte, short, int, long, float, double, char, boolean.
//An object is a collection of data that provides a set of methods.
//String is an object.
```

9.1 Characters, type char.

```
fruit = "banana";
letter = fruit.charAt(0);
out.println(letter);
out.println(letter == 'a');
letter = fruit.charAt(5);
out.println(letter);
out.println(letter == 'a');
out.println('\'');
```

9.1 Characters, type char.

```
//The increment and decrement operators work with characters.
//In Unicode, each character is represented by a code unit.
//The code units for uppercase Latin letters run from 65 to 90.
//The code units for lowercase Latin letters run from 97 to 122.
out.println("Roman alphabet: ");
for (letter = 'A'; letter <= 'z'; letter++) {
        out.print(letter);
}
out.print("\n");
```

9.1 Characters, type char.

```
//The code units for uppercase Greek letters run from 913 to 937.
//The code units for lowercase Greek letters run from 945 to 969.
out.println("Greek alphabet");
for (n = 913; n <= 969; n++) {
        out.print((char)n);
}
out.print("\n");
out.println();
```

9.2 Strings are immutable

```
out.println("9.2 Strings are immutable");
//If once created, an object cannot be modified, the object is immutable.
//Strings are immutable by design.
name = "Alan Turing";
upperName = name.toUpperCase();
//toUpperCase and toLowerCase methods create new strings. They do not change
//the string "Alan Turing".
out.println(upperName);
out.println(name);
```

9.2 Strings are immutable

```
name = name.replace("Turing", "Dubos");
out.println(name);
//We assigned a new string "Alan Dubos" to name, replacing the old string
//"Alan Turing".
out.println();
```

9.3 String traversal

```
out.println("9.3 String traversal");
for (n = 0; n < fruit.length(); n++) {
        letter = fruit.charAt(n);
        out.println(letter);
}
//Strings provide the length method that returns the number of characters in
//the String.
length = fruit.length();
out.print("The last letter of the word \"banana\" is: ");
out.println(fruit.charAt(length -1));
out.println();
```

9.3 String traversal

```
out.print("Please type your name: ");
name = in.nextLine();
reName = reverse(name);
out.println("The reverse of your name is: " + reName);
out.println();
//Write the method reverse(a) to reverse the String a.
```

9.3 String traversal

```
public static String reverse (String name) {






}
```

9.3 String traversal

```
public static String reverse (String name) {
    String reverse = "";
    for (int n = name.length() - 1; n >=0; n--) {
        reverse = reverse + name.charAt(n);
    }
    return reverse;
}
```

9.4 Substrings

```
out.println("9.4 Substrings");
//The substring method returns a new string that copies letters from an
//existing string, starting at the given index.
out.println(fruit.substring(0));
out.println(fruit.substring(6));
//Starting at the first given index and stopping before the second.
out.println(fruit.substring(0,3));
out.println(fruit.substring(5,5));
out.println(fruit.substring(5,6));
out.println(fruit.substring(6,6));
//out.println(fruit.substring(5,7));
out.println();
```

9.5 The indexOf method

```
out.println("9.5 The indexOf method");
out.println("In the String \"banana\",");
index = fruit.indexOf('a');
//It returns the index of the first appearance.
out.println(index);

index = fruit.indexOf('a', 2);
out.println(index);
//Starting at index 2 and finds the next 'a', which is at index 3.
```

9.5 The indexOf method

```
index = fruit.indexOf('x');
out.println(index);
//If the character does not appear in the string, indexOf returns -1.

index = fruit.indexOf("nan");
out.println(index);
```

9.6 String comparison

```
out.println("9.6 String comparison");
//The == operator checks whether two variables refer to the same object.
//If you give it two different strings that contain the same letters,
//it yields false.
String name1 = new String ("Alan Turing");
String name2 = new String ("Alan Turing");
if (name1 == name2) {
        out.println("The names are the same.");
} else {
        out.println("The names are different.");
}
```

9.6 String comparison

```
//The equals method returns true if the strings contain the same characters.
if (name1.equals(name2)) {
        out.println("The names are the same.");
} else {
        out.println("The names are different.");
}

name = "Alan Turing";
out.println("Hello, my name is Alan Turing. What's your name, please?");
reName = in.nextLine();
```

9.6 String comparison

```
diff = name.compareTo(reName);
if (diff == 0) {
    out.println("Our names are the same.");
} else {
    if (diff < 0) {
        out.println("My name comes before yours.");
    } else {
        out.println("My name comes after yours.");
    }
}
out.println();
```

9.6 String comparison

//If the first string comes first in the alphabet, the difference is negative.

//If the strings are equal, their difference is zero.

9.7 String formatting

```
out.println("9.7 String formatting");
//With the inputs hour = 19, minute = 5, we want the method returns
//07 : 05 PM
out.print("Please type the hour: ");
hour = in.nextInt();
out.print("and the minute: ");
minute = in.nextInt();
out.print("The time is ");
out.println(time(hour, minute));
out.println();
```

9.7 String formatting

```
public static String time(int hour, int minute) {
    String ampm;
    if (hour < 12) {
        ampm = "AM";
        if (hour == 0) {
            hour = 12;
        }
    } else {
        ampm = "PM";
        hour = hour - 12;
    }
    return String.format("%02d:%02d %s", hour, minute, ampm);
    //String.format creates a new string, but does not display anything.
}
```

9.8 Wrapper classes

```
out.println("9.8 Wrapper classes");
//Primitive data types do not provide methods. But for each primitive type, there is a
//corresponding class in the Java library, called a wrapper class.
//int has Integer. boolean has Boolean. long has Long. double has Double.
//Wrapper classes provide methods for converting strings to other types.
```

9.8 Wrapper classes

```
out.println(Integer.parseInt("12345"));
out.println(Boolean.parseBoolean("True"));
out.println(Boolean.parseBoolean("true"));
out.println(Boolean.parseBoolean("TRUE"));
out.println(Boolean.parseBoolean("1"));
out.println();

//They also provide toString, which returns a string representation of a value.
out.println(Integer.toString(12345));
out.println(Boolean.toString(true));
out.println();
```

9.9 Command-line arguments

```
out.println("9.9 Command-line arguments");
//See Max.java
```

9.9 Command-line arguments

```
//Chap 9: section 9.9 Command-line arguments
public class Max {
        public static void main (String[] args) {
                int max = Integer.MIN_VALUE;
                for (int i = 0; i < args.length; i++) {
                        int value = Integer.parseInt(args[i]);
                        if (value > max) {
                                max = value;
                        }
                }
                System.out.println("The max is " + max);
        }
}
```

9.9 Command-line arguments

//Find Max.java in the command-line interface, for example:
//D:\Dubos\eclipse-workspace\ThinkJava\src

//First, go to C:\>. Type: **cd **
//or go to D:\>. Type D:
//then type: cd\Dubos\eclipse-workspace\ThinkJava\src
//Show a list of all the files and subprojects. Type: dir

//Compile Max.java. Type: javac Max.java
//Then you will find a new file Max.class in the src folder.
//Max.class contains the byte code.

9.9 Command-line arguments

```
//Execute the program, type: java Max
//The result will be: The max is -2147483648.
//-2147483648 is the minimum value that int handles.
//Since args is empty, the value of max is same as its initial Integer.MIN_VALUE

//If you provide additional values, they will be passed as arguments into Max.
//Type: java Max 1 2 3 -8
//The result will be: The max is 3.
```