Programmierung

Midterm Test (Winter Term 2015/16)

December 16th, 2015

1 Hints

- **Please wait with turning this page until you are told to do so.**
- **Rules for final exam:** You are not allowed to use anything other than blank paper and something to write with. In particular, you are not allowed to use the book, any printouts, or electronic devices (which includes phones and watches). **If you have any of these with you during the final exam, you risk a failing grade.**
- Your midterm will not be graded and will not influence your final grade in any way. The midterm is merely a tool for you to assess where you stand, and how to prepare effectively for the final exam.
- You will have 30 minutes to work on the assignments. Afterwards, we will talk about what the correct answers would have been and will ask your feedback.
- Most questions require just a few words or a short sentence as an answer. All answers should fit into the space provided.
- If you have difficulties answering one question, you should quickly move on to the next question. Try your best to show your knowledge. At least during the first pass through the exam, focus on the questions you can answer, not on those you cannot answer; if you still have time, revisit the problems you could not solve right-away. **Don't worry if you cannot answer all questions in the provided time; you do not need a perfect score to pass the exam.**

2 Assignments

Assignment 1 What are essential properties of an algorithm?

Assignment 2 What is the difference between a class and an object?

Assignment 3 What are the two attributes that define a data type?

Assignment 4 Consider the following piece of Java code. Annotate each of the highlighted parts of code with the corresponding term or terms from the list of terms below by writing the appropriate number(s) into the respective circle.

1. Argument
2. Assignment
3. Block
4. Boolean expression
5. Class
6. Comment
7. Constant
8. Constructor
9. Expression
10. Integer expression
11. Iterative statement
12. Javadoc comment
13. Method call
14. Object
15. Package
16. Parameter
17. Return statement
18. Return type
19. Statement
20. String
21. Term
22. Type
23. Variable
24. Visibility

```
import acm.program.ConsoleProgram;

public class BoundsChecking extends ConsoleProgram {
        private static final int LOWER_BOUND = 23;
        private static final int UPPER_BOUND = 42;

        public void run() {
                int input = readBoundedInt("Enter a number between "
                                + LOWER_BOUND + " and " + UPPER_BOUND + ".",
                                LOWER_BOUND, UPPER_BOUND);
                println(input);
        }

        /**
         * Reads an integer from the console and only accepts it if it falls into
         * the given bounds.
         *
         * @param prompt
         *            the text displayed to the user when prompting for input.
         * @param lower
         *            the lower accepted bound for the input.
         * @param upper
         *            the upper accepted bound for the input.
         * @return a number {@code >= lower} and {@code <= upper}.
         */
        private int readBoundedInt(String prompt, int lower, int upper) {
                int input = readInt(prompt);

                // Prompt user for input again while it's invalid
                while (input < lower || input > upper) {
                        println("The input must be between " + lower
                                + " and " + upper + ".");
                        input = readInt(prompt);
                }

                return input;
        }
}
```

Assignment 5 What is an expression?

Assignment 6 What `for` loop control line would you use in each of the following situations:

1. Counting from 1 to 10.
2. Counting by fives starting at 0 as long as the number has at most three digits.

Assignment 7 What is a scope?

Assignment 8 A progress bar visualizes how much of a given piece of work has already been done. The file copy dialog is a good example. The following piece of code is supposed to draw two progress bars. The percentage of work completed is the same for both calls to drawProgressBar(...), so the two bars should look the same. However, we get the following output:

```
|################################                    |
|#                                                   |
```

Explain why this happens and fix the problem in the code.

```
import acm.program.ConsoleProgram;

public class ProgressBar extends ConsoleProgram {
    public void run() {
        drawProgressBar(50, 30, 50);    // Call 1
        drawProgressBar(50, 300, 500);  // Call 2
    }

    /**
     * Draws a progress bar to the console.
     *
     * @param size
     *            the width of the progress bar, measured in the number of
     *            characters used to actually draw the bar.
     * @param part
     *            how much progress has been made so far.
     * @param whole
     *            how much progress needs to be made to fill up the whole bar.
     */
    private void drawProgressBar(int size, int part, int whole) {
        // Draw left boundary
        print("|");

        // Draw the progress bar itself
        for (int i = 0; i < size; i++) {
            print((i <= size / whole * part) ? "#": " ");
        }

        // Draw right boundary
        println("|");
    }
}
```

Assignment 9 What is a predicate method?

Assignment 10 What role do methods serve in expressions?

Assignment 11 Describe the function of the `println` method. What is the significance of the letters `ln` at the end of its name?

Assignment 12 What are the two classes of control statements?

Assignment 13 Identify which of the following are legal constants in Java. For the ones that are legal, indicate whether they are integers or floating-point constants.

1. `42.0`
2. `-2,3`
3. `3,141`
4. `13`
5. `1.1X+11`
6. `1.1E+2.2`

Assignment 14 Write a class Diamond in a package called programming.exam.midterm.diamond which is a subclass of ConsoleProgram. When the program is run, it should ask the user for an integer $n \geq 0$. As long as the user enters an invalid value for n, she should be asked again for a valid value. Then, the program should print a diamond shape to the console that has n lines above and below the center line. Split your code into methods as appropriate. You don't have to add import statements to your code.

For $n = 3$, the diamond would look like this:

```
   #
  ###
 #####
#######
 #####
  ###
   #
```