P2 - Exercise hour

Pooja Rani

2021-05-28
Exercise 11 Hints

- Split lines:
  `aTurtleProgram lines`

- Split by whitespace:
  `aLine splitOn: Character space`

- Conditionals:
  `(command = `right') ifTrue: [ turtle right: steps ]`

- Regular expressions:
  `'up 15' matchesRegex: '^(left|right|up|down) \d+'`

- ...
Sample exam questions

- What is the pattern of questions?
- How to approach the questions?
Why do god classes and data classes often occur together?

When should you call super() in a constructor and why?

What is iterative development, and how does it differ from the waterfall model?

What are the advantages of using the Model-View-Controller pattern?
You should be aware with all Object-oriented concepts.
You should know what is the role of each concept.
Fix these JavaDoc comments.

```java
/*
 * The <i>Algorithm</i> defines how a value
 * for a file is computed.
 * It must be sure that multiple calls for the
 * same file results in the same value.
 * The implementing class should implement
 * a useful toString() method.
 */
public interface Algorithm {
    // ...
}
```
Write JavaDoc comments for the given method.

```java
/*
 * *
 * */
public int updateAlgorithm(String name, int left) {
    // ...
}
```
Design By Contracts

/* This method updates the algorithm according to the given parameters */

public int updateAlgorithm(String name, int left) {
    // ...
}
Use correct format to write JavaDoc comments.

```java
/* Updates the algorithm according to the given parameters */
public int updateAlgorithm(String name, int left) {
    // ...
}
```
Use tags to write JavaDoc comments.

/* Updates the algorithm according to given parameters
   * @param name ..
   * @param left ..
   * @returns ..
   * @throws ..
   */

public int updateAlgorithm(String name, int left) {
   // ...
}
Write Dbc for the following methods.

```java
public int updateAlgorithm(String name, int left) {
    // ...
}
```
Write Dbc for the following methods.

```java
/* Summary ...
 * ..... 
 * precondition name must not be null.
 * precondition left must be positive.
 * poscondition
 */

public int updateAlgorithm(String name, int left) {
  // ...
}
```
Design By Contracts

Make sure to check the pre or post conditions for the method.

```java
/* Summary of the method
 * ..... 
 * @precondition name must not be null.
 * @precondition left must be positive.
 * @postcondition 
 */

public int updateAlgorithm(String name, int left) {

    //precondition
    
    this.name = name;
    this.position = this.currentPosition + left;
    
    ... 

    //postcondition
}
```
Design By Contracts

Make sure to check the pre or post conditions for the method.

```java
/* Summary of the method
 * ..... 
 * @precondition name must not be null. 
 * @precondition left must be positive. 
 * @postcondition 
 */

public int updateAlgorithm(String name, int left) {

    assert (name != null)

    this.name = name;
    this.position = this.currentPositiion + left;

    ...

    //postcondition
}
```
**Explain** the observer pattern on an example use case of your choice. Include the following in your answer:

- Provide example code.
- Provide an UML diagram of the classes involved.
- State one advantages and one disadvantage of using the Observer pattern to implement a GUI. Use less than 100 words.
Identify the design pattern from the code snippet.

- Explain the pattern.
- Provide an UML diagram of the classes involved.
Modify the existing code of a given design pattern, for example, example code is provided for the factory pattern, add a new object in the existing code.

- Write the necessary code for adding the object.
- Provide an UML diagram of the classes involved.
Design Patterns

You should be able to do this for all the patterns from the lecture and covered in the exercises, for example, adapter, proxy, observer, null object, composite, command, chain of responsibility.. (and more!)
Write a JUnit test that verifies that line 10 works as expected.

```java
public class Spreadsheet {
    private int[][] contents;
    private int rows;
    private int cols;

    /** JavaDoc omitted */
    public void setCellValue(int row, int col, int value){
        if (row < 0 || row > this.rows-1) {
            throw new IllegalArgumentException();
        }
        if (col < 0 || col > this.cols-1) {
            throw new IllegalArgumentException();
        }
        this.contents[row][col] = value;
    }
}
```
Write a JUnit test that verifies that line 10 works as expected.

```java
1. public class SpreadsheetTest {
2. 
3. 
4.   public void testCellValue(){
5.     Spreadsheet spreadsheet = new Spreadsheet();
6.     spreadsheet.setCellValue (..) //cover line 9-7
7.   }
8. }
```
Explain what the following Smalltalk code result into and why?

```smalltalk
rows: rows columns: columns tabulate: aBlock
  | a i |
  a := Array new: rows*columns
  i := 0.
  1 to: rows do: [:row |
    1 to: columns do: [:column |
      a at: (i := i+1) put:
        (aBlock value: row value: column) ] ] ].
^ a
```
This is just a selection of topics.

Everything that was covered in the lectures and exercises can appear in the exam.
Final Remarks

▶ Check whether you got the Testat.
▶ The exam takes place on Wednesday, 9 June, 10:00–12:00 (You get 10 minutes to clarify questions, 100 minutes to solve and 10 minutes to send your solutions via email!).
▶ The exam will take place online via Zoom. Make sure you have zoom installed.
▶ You would need to send solution via the google forum. Make sure you have a google account.