CS 149 Unit: Methods

Specific course level objectives:

* Understand and use appropriate computer programming terminology and concepts.
* Develop clear and correct algorithms to solve problems on the computer.
* Use appropriate software testing tools and procedures to thoroughly test programs.

Unit level objectives:

* Differentiate between void and value returning methods
* Break a large program into several smaller methods
* Draw a stack diagram that illustrates method calls
* Define methods and call them in a program
* Use separate compilation of a “main” class and a “helper” class.
* Define and use correctly: parameter, invocation, pass by value, return type, return statement, method header
* Successfully trace a program with methods and method calls.
* Test methods using the JUnit testing suite.

Method unit outline

Learning activities

Assessment

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Classroom Friday**Introduction to methods. Worksheet. Activity 05B |  | **Lab Monday**Introduction to Methods and Command Line compilation.Formative assessment |  | **Lab Wednesday**Testing methods: Introduction to JUnit testing.Formative assessment |  | **Midterm questions pertinent to methods****-------------****Method PA** |
| Watch tutorial about methodsRead chapter 5.1-5.4 paying particular attention to the Checkpoints. |  | Finish activity if not done during class. Bring “program” to class. |  | Finish Monday lab. View JUnit Testing tutorial (TBB) |  | Finish Wednesday lab.^^^^^^^^Review for midterm |  |

Sample assessment questions:



1. (T/F) A variable declared inside of a method can be used (seen) by the calling method. (U)
2. (T/F) Parameters can be declared only with primitive data types. (U)
3. Methods are commonly used to (U)

|  |  |
| --- | --- |
| a. | increase the amount of documentation we have to write |
| b. | separate code that we can execute multiple times |
| c. | simplify complex code |
| d. | both a and b |
| e. | both b and c |

***Part b*** ***(10 points)*** –Write **ALL** of the line numbers on which the specific features occur in the code, and only **ONE** actual example of the feature from the code. (Ap)

|  |  |  |
| --- | --- | --- |
| **Feature** | **ALL** line numbers | **One** example |
| Method invocation (call)  |  |  |
| Primitive variable declaration |  |  |
| Method return type |  |  |
| Boolean expression |  |  |
| Argument (actual parameter) |  |  |
| Parameter (formal parameter) |  |  |

***Part c*** ***(5 points)*** –If you were designing full coverage tests for the *adjust* method, what tests would you include? Show the parameter values and the expected output of the method for each test (one per line). You may not need all of the rows. (An)

|  |  |
| --- | --- |
| **Parameter value (*number*)**  | **Expected return value** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Question 6** *(20 points)* –Write a complete method based upon the problem specification. The method header is given. *THINK BEFORE YOU WRITE.* (Ap/An)

a. Method calcCost will calculate the costs of specialty coffee. Coffee is sold in three sizes, 8oz, 12oz, and 24oz. 8oz coffee costs $1.50, 12oz $2.25, and 24oz $3.00. Flavor “shots” can be added at the price of $0.50 each for 8oz and 12oz sizes. “Shots” are free for the 24oz size. If a parameter value is invalid (size not 8, 12, or 24, or shots < 0), the method should return -1.

public static double calcCost(int ounces, int shots)

{

}

b. MethodenterPositiveDouble returns a positive double number entered by the user. The method should display the given prompt, and if the user enters a positive number, input and return it. If the value entered is not a number or is not positive, the method should repeat the prompt and input until a valid positive number is entered, and then return it.

public static double enterPositiveDouble(String prompt)

{

}

REFERENCE FOR QUESTIONS ABOVE

|  |  |
| --- | --- |
|  | public class Exam2Trace |
|  | { |
|  |  public static void main(String args[]) |
|  |  { |
|  |  int count; |
|  |  count = 1; |
|  |   |
|  |  do |
|  |  { |
|  |  count = count + adjust(count \* 2); |
|  |   |
|  |  } while (count <= 15);  |
|  |   |
|  |  System.out.printf("Count is %d.\n", count); |
|  |  } |
|  |  |
|  |  public static int adjust(int number) |
|  |  { |
|  |  int value; |
|  |  if (number <= 0) |
|  |  { |
|  |  value = 5; |
|  |  } |
|  |  else if (number % 2 == 1) |
|  |  { |
|  |  value = number / 2; |
|  |  } |
|  |  else |
|  |  { |
|  |  value = number + 1; |
|  |  } |
|  |   |
|  |  System.out.printf("Value is %d. \n", value); |
|  |   |
|  |  return value; |
|  |  } |
|  |  } |