CSC 243
Computer Programming I
J.Khalife
Fall2011

Exam Preparation - B

For exercises 1 to 13, write code segments that will perform the specified action. Assume that all variables have already been declared and given values.

1. Print "Hurrah!" if sum is evenly divisible by count.

2. Increment the integer variable total if total is zero and decrement total otherwise.

3. Print "num is zero", "num is negative", or "num is positive" as appropriate, based on the current value of num.

4. Print "num is zero", "num is even", or "num is odd" as appropriate based on the current value of num.

5. Print "Victory" only if result is greater than or equal to 500 and penalty is equal to zero (use nested ifs).

6. Print "Victory" only if result is greater than or equal to 500 and penalty is equal to zero (use logical operators).

7. Assign the smallest of two integer values num1 and num2 to the variable smallest. (use an if-else statement)

8. Assign the smallest of two integer values num1 and num2 to the variable smallest. (use the conditional operator)

9. Assign the smallest of three integer values num1, num2, and num3 to the variable smallest. (do not use logical operators)

10. Assign the smallest of three integer values num1, num2, and num3 to the variable smallest. (use logical operators)

11. Print "This character is a vowel." if the character stored in the variable letter is a lowercase vowel.

12. Of the two characters stored in the variables ch1 and ch2, print the one which comes later in the Unicode character set.

13. Print "Uppercase", "Lowercase", or "Not a letter" depending on whether the character stored in ch is an uppercase alphabetic character, a lowercase alphabetic character, or not an alphabetic character at all.

14. Print "Equal" if two floating point values stored in val1 and val2 are exactly equal, "Essentially Equal" if they are within 0.0001 of each other, or "Not Equal" otherwise.

For exercises 1 to 10, write code segments that will perform the specified action.

1. Verify that the user enters a positive value. (use a while loop)

2. Verify that the user enters an even value (use a do loop)

3. Read and print values entered by a user until a particular sentinel value is encountered. Do not print the sentinel value. Assume the sentinel value is stored in a constant called SENTINEL.

4. Read values from the user, quitting when a sentinel value of 0 is entered. Compute and print the product of all values entered (excluding the sentinel value).

5. Print the odd numbers between 1 and 100.

6. Print the multiples of 3 from 300 down to 3.

7. Print the numbers between LOW and HIGH that are evenly divisible by four but not by five.

8. Print all of the factors of a value stored in the variable number. Assume the value is positive.

9. Read 10 values from the user and print the lowest and highest value entered.

10. Print a sequence of asterisk characters in the following configuration, continuing for LINES number of asterisks.

```
*
 *
  *
   *
    *
     *
      *
```

Exam Preparation Exercises
Methods SetI

1. Write a method called **average** that accepts three integer parameters and returns their average as a floating point value.

2. Rewrite (Overload) the **average** method of the previous exercise such that if four integers are provided as parameters, the method returns the average of all four.

3. Rewrite (Overload) the **average** method once more to accept five integer parameters and return their average.

4. Write a method called **randomInRange** that accepts two integer parameters representing a range. You may assume that the first parameter is less than or equal to the second, and that both are positive. The method should return a random integer in the specified range.

5. Overload the **randomInRange** method of the previous exercise such that if only one parameter is provided, the range is assumed to be from 1 to that value. You may assume the parameter value is positive.

6. Write a method called **randomColor** that creates and returns a `Color` object that represents a random color. Recall that a `Color` object can be defined by three integer values between 0 and 255 representing the contributions of red, green, and blue (its RGB value).

7. Write a method called **darken** that accepts a `Color` object as a parameter and returns a new Color object that is "darker" than the parameter. Create the darker color using RGB values that are 10 percent less than the original.

8. Write a method called **drawCircle** that draws a circle based on the method's parameters: a `Graphics` object through which to draw the circle, two integer values that define the (x, y) coordinate of the center of the circle, another integer that represents the circle's radius, and a `Color` object that defines the circle's color. The method does not return anything.

9. Overload the **drawCircle** method of the previous exercise such that if the `Color` parameter is not provided, the circle's color will default to black.

10. Overload the **drawCircle** method again such that if the radius is not provided, a random radius in the range 10 to 100 will be used.

11. Overload the **drawCircle** method yet again such that if both the color and radius of the circle are not provided, the color will default to red and the radius will default to 40

Exam Preparation Exercises
Methods Set II

1. Write a method called **isAlpha** that accepts a character parameter and returns true if that character is either an uppercase or lowercase alphabetic letter.

2. Write a method called **floatEquals** that accepts three floating point values as parameters. The method should return true if the first two parameters are essentially equal, within the tolerance of the third parameter.

3. Write a method called **isIsoceles** that accepts three integer parameters that represent the lengths of the sides of a triangle. The method should return true if the triangle is isosceles but not equilateral, meaning that exactly two of the sides have an equal length, and false otherwise.

4. Write a method called **powersOfTwo** that prints the first 10 powers of 2 (starting with 2). The method takes no parameters and doesn't return anything.

5. Write a method called **alarm** that prints the word "Alarm!" multiple times on separate lines. The method should accept an integer parameter that specifies how many times the output line is printed.

6. Write a method called **sum100** that returns the sum of the integers from 1 to 100.

7. Write a method called **sumRange** that accepts two integer parameters that represent a range. You may assume the first parameter is less than or equal to the second. The method should return the sum of the integers in that range.

8. Write a method called **countA** that accepts a String parameter and returns the number of times the letter 'A' is found in the string.

9. Write a method called **multiConcat** that takes a String and an integer as parameters, and returns a String that is the parameter string concatenated with itself n number of times (where n is the second parameter). For example, if the parameters are "hi" and 4, the return value is "hihihihi".

10. Overload the **multiConcat** method from the previous example such that if the integer parameter is not provided, the method returns the string concatenated with itself. For example, if the parameter is "test" the return value is "testtest".

11. Write a method called **validate** that accepts three integer parameters. The first two parameters represent a range, and the purpose of the method is to verify that the value of the third parameter is in that range. You may assume that the first parameter is less than or equal to the second. If the third parameter is not in the specified range, the method should prompt the user and read a new value. This new value should be tested for validity as well. The method should only return to the calling method once a valid value has been obtained, and it should return the valid value.

12. Write a method called **reverse** that accepts a String as a parameter and returns a String that contains the characters of the parameter in reverse order. Note: there is actually a method in the String class that performs this operation, but for the sake of this exercise you will write your own.

13. Write a Method ***Prime*** that determines whether or not an integer ***N*** is a prime Number.