Bluegrass Community and Technical College

Programming Requirements Document

Sleeping Your Life Away

NARRATIVE DESCRIPTION

Estimate the amount of time a person has slept in their life (roughly).

Write a Java application that will input a person's name, age, and a typical number of hours he/she sleeps per night. Valid input could be:

```
Joe Smith
 28
 7.5
```

From that data, calculate:

1. The number of hours the person has slept during his/her life
   - age * 365 * hours slept each night
2. The number of days the person has slept during his/her life
   - Answer from #1 divided by 24
3. The number of years the person has slept during his/her life
   - Answer from #2 divided by 365

NOTE: You are estimating the values above. For example, your age is based upon your last birthday. Do not worry about days since a person's last birthday.

NEW CONCEPTS ASSESSED AND ILLUSTRATED

- Convert a narrative description of a problem into Java code that solves the problem. For example:
  1. Read the narrative description and list all input values required to solve the problem. Create appropriate variables for the input values.
  2. Read the narrative description and list all literal values. For this assignment, the literals would be 24 and 365. Create appropriate constants for the literal values you need to use. Once a constant is declared, use the name of the constant in your calculations instead of the literal values.
  3. Read the narrative description and list all output values required to solve the problem. For example, the number of hours the person has slept their entire life is one required output. Create appropriate variables for the output values.
  4. Read the narrative description and list all calculations and processing required to use the input values to produce the output values. It is best to plan solutions on paper before attempting to key into a program. It will same time in the long run.
  5. Convert items 1-4 into Java code. Compile, debug, and test the solution.

- Make sure you select the correct data type for variables, constants and literals. Make sure you use floating-point and integer values correctly.
- Make sure the identifier names (for variables and constants) are meaningful names. For example, X is not a meaningful name.
- Add internal documentation (comments) as you code. Before you turn in your "completed" work, make sure your program is completely documented.

SOFTWARE REQUIREMENTS (THINGS TO CHECK BEFORE SUBMITTING YOUR WORK)

☐ Select the correct data type for each variable and constant.

☐ Use constants for all numeric literals (such as 24 and 365)

☐ Input the user's name, age, and normal hours slept each night.

☐ For simplicity for the user, input only 1 value at a time. Do not require the user to key multiple items at once.

☐ Output the 3 items mentioned in the narrative.

☐ All input prompts should be clear to the user as to what they are to key.

☐ All output displayed should be clear to the user as to what they are viewing as the results. Make sure you use whitespace to help with readability.

☐ Include a comment block at the top of your program to include
  - Your name
  - Date you created or submitted the program
  - Instructor
  - Class and
  - Purpose of this program

  This is standard for every assignment going forward even if not listed as a specific requirement.

☐ Use the ***Programming Standards*** document from Module 1 to apply documentation standards correctly (such as blocked comments, no comments at the end of lines of code, etc.) Document each section of code. This is standard for every assignment going forward even if not listed as a specific requirement.

SECURITY CONSIDERATIONS

Selecting incorrect data types can have security implications, which we will soon learn about. It is very important that you select the correct data type (and size) for each variable and constant you use. We do not want to waste memory and yet we must insure the data type is large enough to hold the data to be stored in memory. This section in your assignments is simply to remind you to consider security when you design programs.