Lab Workbook 4

Software Development HDSWD
Selection Statements
18/05/12

Jonathan Meaney

For each problem make sure to use the same steps we have been using so far!

1. **Problem Definition**
    a. **Define the problem in your own words**
    b. **Read the problem and determine what it is asking you to do. Highligh the keywords**
    c. **Describe the features of the application and what its goal is**

2. **Design - Overall Plan**
    a. **What objects and classes are you going to use**
    b. **What types of data will I be working with**
    c. **What is the input**
    d. **What processing will be done**
    e. **What will the output be**
    f. **What will be the components of the class be**
        i. **What data members do you need**
        ii. **What will your constructor do (default initialization, initialize to different values using overloaded constructors)**
        iii. **What methods will you need (getters and setters for data members, what behaviour is needed, what will it do)**

3. **Implementation**
    a. **Create the classes**
    b. **Declare and create objects**
    c. **Provide input and display output**

4. **Testing**
    a. **Was the output correct?**
    b. **Did the program compile and run successfully**
    c. **Test the application with different values**

Problem 1:

Create a main class called Problem1 containing a main method. Declare an integer variable called score and give it an initial value of 50. Create suitable if statements to compare the value of score to the following values and print a different messages for each one. Make correct use of logical and comparison operators. i.e. (score >= 10 && score < 50). Re run the program with different values for score to test the different outputs.

Score value	Message
>= 10	**You won a free go**
>= 50	**You won a half eaten pie**
>= 100	**You won a luxury trip for two to NCI's basement**
>= 200	**You won a pair of cruise liners**
>= 300	**You won 16 Rolex watches**
>= 400	**You won the largest gold nugget in the world, 1 mile in diameter**
>= 500	**You won the greatest hits of Alanis Morissette deluxe 18 cd box set**

Problem 2:

One million is 10^6 and 1 billion is 10^9. Create a class called Problem2 containing a main method that reads a power of 10 (6, 9, 12, etc.) from the user and displays how big the number is (Million, Billion, etc.), use Javabook to provide the input and output features, create a MainWindow object an InputBox object and a MessageBox object like previous examples and use the getInteger method to take an integer from the user. Display an appropriate message for the input value that has no corresponding word. The table below shows the correspondence between the power of 10 and the word for that number. Use a switch statement to implement the solution to this problem.

Power	Word
6	**Million**
9	**Billion**
12	**Trillion**
15	**Quadrillion**
18	**Quintillion**
21	**Sextillion**
30	**Nonillion**
100	**Googol**

Problem 3:

Update the Account class from Lab3, (available on moodle in lab 3 solutions under week 4 lab section). A user shouldn't be able to withdraw more money than they have in their account. Update the withdraw method to use if statements to check that the balance is greater than the amount being withdrawn, if it is not display an appropriate error message. Create a new or reuse the main class and main method from the lab 3 solutions for this application. In the main method for this class change the amounts being withdrawn and the balance in the account to test the functionality of the withdraw method.

Problem 4:

Update the car class from lab 3.

- **Give the class a new data member called petrolLevel**
  - **This data member is of type double**
  - **Update the constructors to initialize the petrolLevel for new objects**
  - **Create setter and getter methods for petrolLevel**
  - **Update the printDetails method for petrolLevel information**
- **Create a method called drive which takes an integer value as a parameter called distance and returns nothing.**
  - **The distance is in miles.**
- **When the car drives it uses petrol so the petrolLevel should decrease by 1.5 for every mile the the car drives**
- **The car can only drive the complete distance specified**
- **If the car has no petrol then it cannot drive. Use if statements to check the level of petrol is greater than the petrol needed to drive the specified distance and if it is then decrease the petrolLevel by the appropriate amount. If there is no petrol then display an appropriate error message**
- **Since the car is able to drive now the milesDriven value should change. If the car can complete the distance specified increase the milesDriven instance variable by the appropriate amount**

Problem 5:

Write a program that replies either "Leap Year" or "Not a Leap Year", given a year.

1. **It is a leap year if the year is divisible by 4 but not by 100 (for example, 1796 is a leap year because it is divisible by 4 but not by 100).**
2. **A year that is divisible by both 4 and 100 is a leap year if it is also divisible by 400 (for example, 2000 is a leap year, but 1800 is not).**

Use the modulus operator to check if two numbers divide, if they do they leave a remainder of 0. Use logical operators to perform multiple checks inside the one if statement Boolean expression. You will also need to use a nested if statement in this application.