Lesson 44: Iteration Revisited #1 (W13D2)

Balboa High School

Michael Ferraro

November 10, 2015

Pre-Do Now

- PS #6 paper form complete? Prepare it for pickup. Sign-offs may be available later in the period.
- Create a new workspace folder called `workspace2`.
- Point Eclipse to the new workspace directory
- Create project `Lesson44` to contain the Do Now.

proceed to next slide →

Do Now

Finish the program so that it prints a triangle with a base of `n` *s.

```
public class PrintTriangle {

    public static void main(String[] args) {
        printTriangle(4);
    }

    public static void printTriangle(int n) {
        /* ex:  n = 3
         *
         *     *
         *     **
         *     ***
         */
    }
}          //FINISHED EARLY?  GO TO THE NEXT SLIDE!
```

Do Now Extension

Alter the `main()` so that the output results in the following by having `printTriangle(...)` called on only one line:

```
*

*
**

*
**
***

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

Aim

Students will learn about the `do-while()` variant of the `while()` statement, how to break out of loops before the stopping condition is reached, and write a program to calculate compound interest.

do-while()

- Sometimes, when a `while()` loop is used, we want to run the code block at least once, regardless of whether the stopping condition is true.

`do-while()`

- Sometimes, when a `while()` loop is used, we want to run the code block at least once, regardless of whether the stopping condition is true.
- Example: The ATM Welcome screen
  - No matter what, print out the menu of options the first time the method runs.
  - Continue repeating that menu after a user runs some action *unless* that user chooses to quit

do-while()

```
//excerpt from AtmWelcome.java:

boolean quitNow = false;

while (!quitNow) {
    printWelcomeScreen();
    c = getChoice();
    switch (c) {
        case 'c':
            checkBalances();
            break;
        ///// other choices...
        case 'q':
            quitNow = true;
            break;
        default:
            System.out.println("INVALID CHOICE.");
    }
}
```

do-while()

Rewriting in positive terms (use a default value of `true` instead of `false`)...

do-while()

```
//excerpt from AtmWelcome.java:

boolean showMenu = true;

while ( showMenu ) {
    printWelcomeScreen();
    c = getChoice();
    switch (c) {
        case 'c':
            checkBalances();
            break;
        ///// other choices...
        case 'q':
            showMenu = false;
            break;
        default:
            System.out.println("INVALID CHOICE.");
    }
}
```

do-while()

- `boolean showMenu` acts as a *flag*

do-while()

- `boolean showMenu` acts as a *flag*
  - It's initially set to `true`
  - `while( showMenu )` will run the code in the block the first time, guaranteed
  - If a user chooses q, the flag is set to `false`, causing `while( showMenu )` to terminate the loop the next time

do-while()

- `boolean showMenu` acts as a *flag*
  - It's initially set to `true`
  - `while( showMenu )` will run the code in the block the first time, guaranteed
  - If a user chooses q, the flag is set to `false`, causing `while( showMenu )` to terminate the loop the next time
- Better way: Use `do-while()`

do-while()

- `boolean showMenu` acts as a *flag*
  - It's initially set to `true`
  - `while( showMenu )` will run the code in the block the first time, guaranteed
  - If a user chooses q, the flag is set to `false`, causing `while( showMenu )` to terminate the loop the next time
- Better way: Use `do-while()`
  - Arguably not better! Some programmers avoid `do-while()` and use the `while()` approach you've already seen.

do-while()

- `boolean showMenu` acts as a *flag*
  - It's initially set to `true`
  - `while( showMenu )` will run the code in the block the first time, guaranteed
  - If a user chooses q, the flag is set to `false`, causing `while( showMenu )` to terminate the loop the next time
- Better way: Use `do-while()`
  - Arguably not better! Some programmers avoid `do-while()` and use the `while()` approach you've already seen.
  - *Do* _____, *repeat while (* _____ *)...*

do-while()

```
//updated to use do-while():

boolean showMenu = true;

do {
    printWelcomeScreen();
    c = getChoice();
    switch (c) {
        case 'c':
            checkBalances();
            break;
        ///// other choices...
        case 'q':
            showMenu = false;
            break;
        default:
            System.out.println("INVALID CHOICE.");
    }
} while( showMenu );
```

Breaking out of a loop

There are two ways to exit a loop aside from the usual way, when the stopping condition is reached

- `return`
- `break`

Breaking out with `return`

- Particularly useful when doing an *array walk*, so we'll wait until then for a worthwhile in-class example.
- In the meantime, make sure you read the `isPrime()` example on Litvin p190 for how to use `return` to break out of a loop.

Breaking out with `break`

Consider when you might run a `while()` or `for()` loop and find a result before it's done running. You have two options:

Breaking out with `break`

Consider when you might run a `while()` or `for()` loop and find a result before it's done running. You have two options:

- Allow the loop to continue running,
  - consuming more CPU cycles than necessary, taking more time
  - possibly making more changes to your data, which might be bad!

Breaking out with `break`

Consider when you might run a `while()` or `for()` loop and find a result before it's done running. You have two options:

- Allow the loop to continue running,
  - consuming more CPU cycles than necessary, taking more time
  - possibly making more changes to your data, which might be bad!
- Exit the loop, even though the stopping condition hasn't been reached

Breaking out with `break`

Example: You deposit $5600 into a fund that has a guaranteed 5.1% annual rate of return

Breaking out with `break`

Example: You deposit $5600 into a fund that has a guaranteed 5.1% annual rate of return

Year	**Balance in USD**
0	5600
1	?
2	?
3	?
...	...

Breaking out with `break`

Example: You deposit $5600 into a fund that has a guaranteed 5.1% annual rate of return

Year	**Balance in USD**
0	5600
1	?
2	?
3	?
...	...

→ *How do you calculate any given year's balance if you know the balance of the prior year?*

Breaking out with `break`

Your task:

1. Download `CompoundInterest.java` from here, and import into a new project called `Lesson44`.
2. Run the program, getting a sense for how it works.
3. Notice how `balance` has more digits past the decimal than we'd prefer? *Round to the nearest hundredth after calculating a new value for* `balance`.
4. Within the `for()` block, include an `if()` statement that tests whether `balance` has reached or exceeded the `target` amount. If so, print a statement saying that by year `year`, `balance` reached or exceeded `target`.
5. Once the last item works, issue a `break` statement immediately after printing out the first year that `balance` meets/exceeds `target`.

Next. . .

- For those up to a challenge: Write a program that figures out how many half-lives it would take for 1kg of plutonium to have a mass of 0.07kg.
- Start working on PS #7

HW

PS #7, §§1-4, inclusive

PS #6 Sign-Offs

- §**5.4, #2:** Demonstrate `CrapsTest1` with this sequence:
  `2-, 3-, 12-, 7+, 11+, 5, 7-, 5, 4, 5+` [1]
- §**5.4, #3:** Show `Die` test class — need to see the entire range of outcomes (1 through 6)
- §**5.4, #4:** Run `CrapsStats` for 100,000 games
- §**6.2:** Run the `switch-case-break` program for these inputs:
  `5, 3, 0`
- §**7, #4:** Show the `Craps` program running, pause after each roll so the number of dots showing can be counted and checked against the win/loss/point results

[1] + = win, - = loss