Lecture 16

- Covers
  - Constructing loop statements
  - Nested loops
- Reading: Savitch 3.3

Lecture overview

- "Off-By-1" Errors and Some Techniques to Handle it
- Three Forms of Loop Control
- Nested Loops

- Off-by-1 errors & general techniques to handle them

Example revisited

- Problem*
  - An account with the initial balance of $1000
  - The interest rate is 10% per year (compounded yearly)
  - How long will it take for the balance to double itself?

* From Horstmann, Big Java, p. 228

Java solution

```
double initialBalance = 1000;
final double RATE = 0.10;
double targetBalance = 2 * initialBalance;

int years = 0;
balance = initialBalance;

while ( balance < targetBalance )
{
    years ++;
    balance = balance + balance * RATE;
}

System.out.println("Balance doubles after " + years + " years");
```

“Off-by-1” errors

- Example
  - Consider the solution for the previous example
  - Consider the question: Should we start with years = 0 or 1?
- How do we work out the answer?

“Off-by-1” errors

- Off-by-1 errors can be quite difficult to solve

  “Some people try to solve off-by-1 errors by randomly inserting +1 or -1 until the program seems to work”*

- There is no silver bullet

* Horstmann, *Big Java*

“Off-by-1” errors

- But there are useful strategies
  1. Try some test cases
     - Devise some simple test cases, and use the information gained from testing them
  2. Observe relationships among “key” variables
     - Observe some relationships among the “key” variables at various points to reason about the loop’s behaviour
  3. Print out diagnostic messages
     - Print out data, especially from within the loop, to make the loop’s behaviour more visible

Using test cases

- Try the test case in which
  - Initial balance is $1000
  - Interest rate is 10%
- Consider
  - How many times should the loop be repeated?
  - What should be the answer?
  - So, what should be the initial value for years?

Observing relationships

- Consider the relationship between years and balance
  - Before entering the loop
  - Within a repetition of the loop
  - After emerging from the loop
- Draw conclusions

Printing diagnostic messages

- Consider printing out information about years and balance
  - Before the looping
  - During the looping (generally very useful)
  - After the looping
- See how this may help us understand the behaviour of the loop

- Three forms of loop control

Which one?

- Which type of loop should I use?
  - while
  - do…while
  - for

Loop equivalences

```
for (initialisation; condition; update action)
{
    body
}
```

```
initialisation
while (condition)
{
    body
    update action
}
```

```
initialisation
if (condition)
{
    do
    {
        body
        update action
    } while (condition);
}
```

Things to consider

- Body of loop
- Initialising statements
- Conditions for ending the loop
- Update action

Three forms of loop control

- Count controlled loops
- Event controlled loops
- Count and event control loops

Count controlled loops

- Perform some action a set number of times
- The for loop is most suitable
- Examples
  - Sum the first 100 integers
  - Calculate the wages for all 34 employees in a company

Count controlled

- Calculate wages for 34 employees in a company

```
for (int i = 1; i <= 34; ++i)
{
    // calculate wages for ith employee
}
```

  - Aside: what is the scope of the variable i?
  - Symmetric bounds

Count controlled

- When repeating a loop a fixed number of times, try to use an inclusive lower bound and an exclusive upper bound (ignore this advice)

```
for (int i = 1; i < 35; ++i)
{
    // calculate wages for ith employee
}
```

- 35 - 1 = 34 times in the loop
  - Asymmetric bounds

Class exercise

- Problem

  Calculate the series

  1/1 + 1/2 + 1/3 + 1/4 + ... + 1/n

  where n is entered by the user

Solution

```
double total = 0.0;
System.out.println("Please enter n: ");
int n = keyboard.nextInt( );

for (int i = 1; i <= n; ++i)
{
  total = total + 1.0 / i;
}
System.out.println("The result is " + total);
```

* *It is better here to use a symmetric loop rather than force an exclusive upper bound*

Event controlled loops

- Repeat some action until a certain event occurs
- while and do...while are most appropriate
- Examples
  - Read input until -1 is entered
  - Calculate wages for employees in a company (without knowing in advance how many employees will be processed)

Event controlled

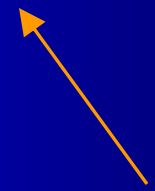
- Add integer inputs until -1 is entered

```
total = 0;
next = keyboard.nextInt( );
while (next != -1)
{
    total += next;
    next = keyboard.nextInt( );
}
```

Event controlled (show the equivalence of while, do-while, for)

- Using a for loop

```
for (total = 0, next = keyboard.nextInt( );
      next != -1;
      next = keyboard.nextInt( );
{
       total += next;
}
```



Not recommended

Another example

- Say “Hello” until the user wants to stop

```
char wish;
String wishString = “”;
do
{
    System.out.println("Hello!");
    System.out.println("Want me to say hello again? (y/n)");
    wishString = keyboard.nextLine();
    wish = wishString.charAt(0);
}
while (wish != 'n');
```

Count and event controlled loops

- Combinations of events and counting are used to terminate the loop
- Such termination expressions are based on combinations of conditions
- Example
  - Play a game while the user wants to, but at most play 5 games
- while or do…while loops are most suitable

Count and event controlled

- Play a game while the user wants to, but at most play 5 games

```
count = 0;
answerString = ““;
do
{
    // code to play game;
    count++;
    System.out.println("Do you want to play again? (y/n)");
    answerString = keyboard.nextLine( );
    answer = answerString.charAt(0);
}
while ( (answer != 'n') && (count < 5) );
```

- Nested loop structures

Nested loop structures

- We can place one loop statement inside another, to create nested loop structures
- Nested loops allow us to solve more complex problems than single-level loops
- When writing nested loops, pay attention to the boundary conditions of both the outer and inner loops
- Consider how the outer and inner loops are related

Example - square of stars

- Write a program to display an n x n square, n is entered by the user

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

for n = 5

Algorithm

LOOP FOR row = 1 to n
*   Output n stars on one line*
ENDLOOP

To Output n stars on one line:
LOOP FOR column = 1 to n
*   Output "* "*
ENDLOOP
Output newline

Giving the nested for loops:
LOOP FOR row = 1 to n
*   LOOP FOR column = 1 to n*
*   Output "* "*
*   ENDLOOP*
Output newline
ENDLOOP

Java code

```
System.out.print("Enter size of square: ");
int n = keyboard.nextInt( );

for (int row = 1; row <= n; ++row)
{
    for (int column = 1; column <= n; ++column)
    {
        System.out.print("*  ");
    }
    System.out.println( );
}
```

Example - star triangle 1

- Write a program to print the pattern of n rows of stars

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

for n = 5

Algorithm

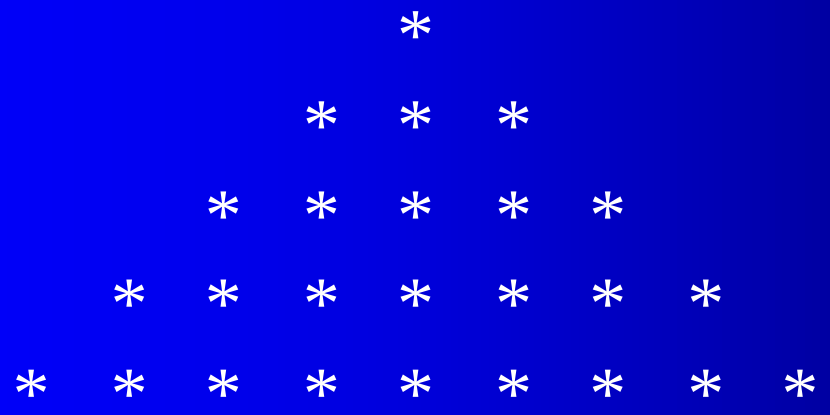
LOOP FOR row = 1 to n
*   Output row stars on one line*
ENDLOOP

To Output row stars on one line:
LOOP FOR column = 1 to row
*   Output "* "*
ENDLOOP
Output newline

Giving the nested for loops:
LOOP FOR row = 1 to n
*   LOOP FOR column = 1 to row*
*   Output "* "*
*   ENDLOOP*
*   Output newline*
ENDLOOP

Example - star triangle 2

- Write a program to print the pattern of n rows of stars



for n = 5

Algorithm

LOOP FOR row = 1 to n

Calculate number of spaces to output

Calculate number of stars to output

LOOP FOR column = 1 to spaces

Output " "

ENDLOOP

LOOP FOR column = 1 to stars

Output " "*

ENDLOOP

Output newline

ENDLOOP

Example - addition table

- Write a program to display the addition table

```
 0  1  2  3  4  5  6  7  8  9 10 11 12
 1  2  3  4  5  6  7  8  9 10 11 12 13
 2  3  4  5  6  7  8  9 10 11 12 13 14
 3  4  5  6  7  8  9 10 11 12 13 14 15
 4  5  6  7  8  9 10 11 12 13 14 15 16
 5  6  7  8  9 10 11 12 13 14 15 16 17
 6  7  8  9 10 11 12 13 14 15 16 17 18
 7  8  9 10 11 12 13 14 15 16 17 18 19
 8  9 10 11 12 13 14 15 16 17 18 19 20
 9 10 11 12 13 14 15 16 17 18 19 20 21
10 11 12 13 14 15 16 17 18 19 20 21 22
11 12 13 14 15 16 17 18 19 20 21 22 23
12 13 14 15 16 17 18 19 20 21 22 23 24
```

Algorithm

LOOP FOR i = 0 to 12

LOOP FOR j = 0 to 12

Output i + j right-aligned, 3 character widths

ENDLOOP

Output newline

ENDLOOP

Java Solution

Class exercise

- Write a loop structure to find all the prime numbers up to a given number that is entered by the user

Solution

Next lecture

- Boolean expressions