Lecture 3

Unicode, Standard I/O, Branching

Introduction to Computer Programming - CST8110

Algonquin College - David Lareau

Fall 2017

School of Advanced Technology
ALGONQUIN COLLEGE

Tutorials and Books

Even though the textbook for this class is a great resource, I've added a list of free alternative books or websites that can help you learn in the Miscellaneous section of our course in Blackboard.

Escape Sequence

Certain characters cannot be typed in a String literal, and need what is known as an escape sequence. We use a backslash to begin the sequence.

```
System.out.println("double quote \"   Awesome");
System.out.println("new line     \n   Awesome");
System.out.println("backslash    \\   Awesome");
```

Warning

- This is especially inconveninent in Windows, where paths are written with backslashes.
- ```
  String path = "C:\\Windows\\System";
  ```
  ```

Escape Sequence

You can use an escape sequence for a single character as well.

```
char normal_char   = 'a';
char double_quote  = '"';
char single_quote  = '\'';
char new_line      = '\n';
char unicode_char2 = 'Ω';      // greek omega
char unicode_char  = '\u03A9'; // unicode
```

Standard Input

We saw last week how to setup a Scanner object to read from the Standard Input.

```
import java.util.Scanner;

public class StandardInput {

  public static void main(String [] args) {
    Scanner scanner = new Scanner(System.in);

    // ask the user something
    System.out.println("Enter a line of text:");
    String line = scanner.nextLine();

    // report
    System.out.println("You wrote:" + line);
  }

}
```

Standard Input

There are other methods aside nextLine() to read specific types.

```
// ask the user for a number
System.out.println("Enter a number:");
int number = scanner.nextInt();

// report
System.out.println("Multiplied by 2 that gives: "
    + (number * 2));
```

- nextInt()
- nextDouble()
- next() to read a single word in a String
- nextLine() to read the rest of the line in a String

Standard Input

Note

- The command line window is responsible for sending the input to your program, and does so on a line by line basis.
- This means that you will have to press Enter for your program to unblock from a Scanner next() function.
- It also means you can send multiple values on the same line.
- This isn't any good for arcade video games.

Demo

- Sending multiple values on the same line.
- Sending the wrong type.
- Input Stream pipe diagram example on the whiteboard.
- Confusion when mixing nextLine() and the other methods.

Boolean

There is a variable type that holds a true or false value.

```
public static void main(String [] args) {
  boolean likesApples = true;
  boolean likesLasagna = false;
}
```

Boolean

There are test operators that resolve to boolean values.

```
public static void main(String [] args) {
  boolean twoIsGreaterThanThree = 2 > 3;

  System.out.println(twoIsGreaterThanThree);

  // What do you think this will print?
  System.out.println(2 == 2); // equals
  System.out.println(2 != 2); // doesn't equal
  System.out.println(2 < 2); // lower than
  System.out.println(2 <= 2); // lower or equal to
  System.out.println((5 % 2) == 1);
  System.out.println(!(2 < 2)); // not
}
```

Boolean

The **not** operator is written with a exclamation mark '!', and inverse the boolean value.

```
public static void main(String [] args) {
  boolean test = 2 < 5;                    // true
  System.out.println(test);                // true
  System.out.println(!test);              // false
  test = !test; // toggles from true to false
  System.out.println(test);               // false
}
```

Branching using if

```
public static void main(String [] args) {
   Scanner scanner = new Scanner(System.in);

   // ask the user their age
   System.out.println("How old are you?");
   int age = scanner.nextInt();

   // if they are old enough, let the user
   // know that is what you wanted.
   if(age >= 13) {
      System.out.println("Perfect.");
   }
}
```

Branching using if

The expression within the round parenthesis of an **if** statement must resolve to a boolean.

```
public static void main(String [] args) {
   if(true) {
   }
   if(false) {
   }
   if(2 + 2) { // compilation error
   }
   if(2 + 2 < 3) {
   }
}
```

Branching using if

An assignation evaluates to the value put in the variable. Be careful. Comparing requires two '='.

```
public static void main(String [] args) {
   int a = 1;
   int b = 1;
   if(a = b) { // compilation error
   }
   if(a == b) { // good: compares a and b
   }
}
```

Branching using if

An assignation evaluates to the value put in the variable. Be careful. Comparing requires two '='.

```
public static void main(String [] args) {
  boolean a = false;
  boolean b = false;
  if(a = b) { // oops, compiles
              // probably not what you meant
  }
  if(a == b) { // good: compares a and b
  }
}
```

Side Demo

- You can chain assignation as in:

```
a = b = false;
```

Branching using if

The block of code after an **if** is skipped if the boolean value resolves to false.

```
public static void main(String [] args) {
  if(2 + 2 >= 4) {
    System.out.println("A"); // always executed
  }
  if(2 + 2 < 4) {
    System.out.println("B"); // never executed
  }
}
```

Side Demo

- You can do a simple statement instead of opening a block after an if.
- You can open a block anywhere for scope management (more on this topic another time).

if/else

You can specify a block of code to execute when the condition resolves to false.

```
int a = scanner.nextInt();
int b = scanner.nextInt();
if(a > b) {
  System.out.println("A is greater");
} else {
  System.out.println("B is greater or equal to A");
}
```

String Comparison

String are objects, and objects are not tested for equality the same way as primitive types are. We must use the equals() method instead of the == operator.

```
String name1 = scanner.nextLine();
String name2 = scanner.nextLine();

// compares addresses to the heap
if(name1 == name2) {
  System.out.println("name1 and name2 are the same
    object."); // never executes
}

// compares the character in the String
if(name1.equals(name2)) {
  System.out.println("name1 and name2 contain the
    same sequence of text.");
}
```

Decision Tree

Demo

- Let's write a program that helps us decide if we should bring our umbrella today.
- Let's write a variant of the 20 questions game called 2 questions.

Normal Indentation

```
public class Indentation {

  public static void main(String [] args) {
    Scanner scanner = new Scanner(System.in);

    // ask the user their age
    System.out.println("How old are you?");
    int age = scanner.nextInt();

    // if they are old enough, let the user
    // know that is what you wanted.
    if(age > 13) {
      System.out.println("Perfect.");
    }
  }

}
```

Non-conventional Indentation

```
public class Indentation
   {

   public static void main(String [] args)
      {
      Scanner scanner = new Scanner(System.in);

      // ask the user their age
      System.out.println("How old are you?");
      int age = scanner.nextInt();

      // if they are old enough, let the user
      // know that is what you wanted.
      if(age > 13)
         {
         System.out.println("Perfect.");
         }
      }
   }
```

No Indentation

```
public class Indentation {
public static void main(String [] args) {
Scanner scanner = new Scanner(System.in);
// ask the user their age
System.out.println("How old are you?");
int age = scanner.nextInt();
// if they are old enough, let the user
// know that is what you wanted.
if(age > 13) {
System.out.println("Perfect.");
}
}
}
```

else if

Normal indentation is a poor choice for an imbricated sequence of if/else/if/else/...

```
if(x == 1) {
    // one
} else {
    if(x == 2) {
        // two
    } else {
        if(x == 3) {
            // three
        } else {
            // not one, two, nor three.
        }
    }
}
```

else if

In a case like this, the convention is not to open the else block with parentheses, and leaving the if on the same line.

```
if(x == 1) {
    // one
} else if(x == 2) {
    // two
} else if(x == 3) {
    // three
} else {
    // not one, two, nor three.
}
```

This is much more readable.

Hybrid Assignment

Demo

- Let's talk about the second hybrid which involves Coding Standards.

Assignment

Demo

- Let's talk about the first assignment, and how the next lab hours will work.

Mini Test

Demo

- Let's talk about the first mini test coming next week.
- The duration is 30 minutes, but can be completed in 10 minutes.
- The test will be held at the end of the second hour of the lecture.
- For students with CAL letters of accomodations, note that I am factoring in extended time for you already.
- If a CAL student still feels the need to schedule the mini test at the CAL center, you can still do so.
- Some CAL letter mention "computer" as a aid, but the exam is on paper (and having a computer with a compiler would make most question very easy). Please let me know if you need a computer for a specific reason, and we can arrange it.