Translating to Java

GEEN163 Introduction to Computer Programming

Translation

- The job of a programmer is to translate a problem description into a computer language.
- You need to be able to convert a problem description into an algorithm and the algorithm into Java.

Many Level Translations

- Requirements
- Design
- Algorithm
- Java
- Machine Language

Input

- Programs usually get some kind of input from the user.
- Input for our programs will usually be from the keyboard.
- More advanced programs can get their input from files, network or Graphical User Interface (GUI).
- Problem descriptions usually use these words for input:
  - read, get, input, ask, request
Java Keyboard Input

- The `Scanner` class is used for input
  ```java
  java.util.Scanner in = new java.util.Scanner(System.in);
  anInt = in.nextInt();
  ```
- The program will wait for the user to enter data.

Java Dialog Box Input

- The `JOptionPane` class can be used for input with a pop up dialog box.
  ```java
  reply = javax.swing.JOptionPane.showInputDialog("Enter something");
  ```
- A box will pop up for every input. The input is always a String which may need to be converted to a double or int.

Output

- Programs generally display some kind of output or you will not know what it did.
- In our class the output has been to the console or pop up boxes.
- More advanced programs output to files, network or GUIs.
- Problem descriptions usually use these words for output:
  ```
  display, write, show, print
  ```

Java Console Output

- The `System.out.println` method can be used for output to the console.
  ```java
  System.out.println("message");
  System.out.print("enter stuff>");
  ```
- The output isn't pretty, but it is easy to do.
Java Dialog Box Output

- The JOptionPane class can be used for output with a pop up dialog box.

```java
javax.swing.JOptionPane.showMessageDialog(null,"Something")
```

- This is useful for outputting a single value.

Outputting Text and Variables

- Frequently you will want to output the value of a variable in the middle of an English sentence.

```java
System.out.println("x is "+avg);
```

- Text is surrounded by quotes. Variables are not.

Decisions

- Most programs will need to make decisions based on a comparison of variables.
- Problem descriptions usually use these words for decisions:

```
when, if, unless
```

Java IF statement

- The if statement in Java allows the programmer to choose different paths of execution.
- Based on a comparison of variable values, the program can do different things.
- An if statement changes the flow of control
If Syntax

\[
\text{if ( Logical Expression )}
\]

Statement

NOTE: Statement can be a single statement or a block.

If Examples

double mph;
mph = in.nextDouble();
if (mph > 55.0) {
    System.out.println("tickets");
} else 
    System.out.println("snail");

If-Then Else Syntax

\[
\text{if ( Logical Expression )}
\]

StatementA

else

StatementB

NOTE: StatementA and StatementB each can be a single statement, or a block.
Translating to C++

if ... else provides two-way selection
between executing one of 2 clauses (the if clause or the else clause)

TRUE FALSE

expression

if clause else clause

Use of Brackets Recommended

if ( Logical Expression ) {
    statements;  "when true"
} else {
    other statements;  "when false"
}

Comparison Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>less than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>less than or equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>greater than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>greater than or equal to</td>
</tr>
<tr>
<td>==</td>
<td>equal to</td>
</tr>
<tr>
<td>!=</td>
<td>not equal to</td>
</tr>
</tbody>
</table>

Boolean Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>not</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>and</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>^</td>
<td>exclusive or</td>
</tr>
</tbody>
</table>
**Convert these to Java**

```java
double quarters; String word; int width;
double cx, cy;
```

- `quarters` is greater than zero
- `width` plus the length of the `word` is greater than 25
- `word` does not contain “DATA”

```
\sqrt{cx^2 + cy^2} \leq r
```

**Possible Solutions**

- `quarters > 0`
- `width + word.length() > 25`
- `word.indexOf("DATA") == -1`
- `Math.sqrt(cx*cx+cy*cy) <= r`

**Repetition**

- Many programs will repeat parts of the program in a loop.
- Problem descriptions usually use these words for loops:
  - `while`, `repeat`, `until`

**Java While Statement**

```java
while ( Expression ) {
    .
    .    // loop body
    .
}
```

NOTE: Loop body can be a single statement or a block.
When the expression is tested and found to be false, the loop is exited and control passes to the statement which follows the loop body.

**WHILE LOOP**

When the expression is tested and found to be false, the loop is exited and control passes to the statement which follows the loop body.

**WHILE LOOP**

Program Logic

```java
while ( Expression ) {
    // the loop body must change the value of a variable in the Expression
}
```

After the loop, you know that the `Expression` is false.

### Variables and Calculations

- Most of our programs will calculate something.
- Many times a program will just move the value of one variable to another.

```java
starttime = now;
count = 1;
```

### Java Variable Declarations

- You have to tell Java what variables your program will be using.
- You have to define the type of the variable

```java
type var1, var2;
```

- where type is int, double, String, etc.

```java
int mythings;
double here2there;
```
Java Assignment Statement

- An assignment statement gives a variable a value;

```java
var1 = 47;
var2 = 23.25 + var3 / 17.0;
```

Translating from Math to Java

```java
diam = \pi r^2
diam = 3.14159265 \times r \times r;

x = \frac{a + b}{c \times d}
x = (a+b)/(c*d);

p = \frac{n \times r \times t}{v}
p = n \times r \times t / v;
```

Integer Division

- Integers can only hold whole numbers
- If division results in a fractional part, the fraction is dropped and the result is just the whole number.

```
8/2 is 4      7 / 3 is 2
1 / 2 is 0      5 / 4 is 1
```

Operator Priority

<table>
<thead>
<tr>
<th>Precedence</th>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Higher</td>
<td>( )</td>
<td>Parenthesis</td>
</tr>
<tr>
<td></td>
<td>+</td>
<td>Positive</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>Negative</td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>Multiplication</td>
</tr>
<tr>
<td></td>
<td>/</td>
<td>Division</td>
</tr>
<tr>
<td></td>
<td>%</td>
<td>Modulus (remainder)</td>
</tr>
<tr>
<td>Lower</td>
<td>+</td>
<td>Addition</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>Subtraction</td>
</tr>
<tr>
<td></td>
<td>=</td>
<td>Assignment</td>
</tr>
</tbody>
</table>
Put the Program in the Template

```
public class Whatever {
    public static void main(String[] a) {
        // variable declarations here
        // The rest of the program here
    }
}
```