Today’s topics

- Parsing
- Java Programming

Reading
- Great Ideas, Chapter 3 & 4

What can Robots do?

- Objects have various methods
  - Services they provide

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>move</td>
<td>Move forward one block. It continues to face the same direction. If the way is blocked by a wall, the robot will “break” into pieces.</td>
</tr>
<tr>
<td>turnLeft</td>
<td>Pivots 90° to the left. The robot remains on the same corner.</td>
</tr>
<tr>
<td>pickThing</td>
<td>Pick up one thing from the corner on which it is standing and add it to its backpack. If there is no thing to pick up, the robot will “break” into pieces.</td>
</tr>
<tr>
<td>putThing</td>
<td>Take one thing out of the robot’s backpack and put it on the corner. If there is no thing in the robot's backpack, the robot will “break” into pieces.</td>
</tr>
<tr>
<td>getAvenue</td>
<td>Returns the avenue that the robot is currently at.</td>
</tr>
<tr>
<td>getStreet</td>
<td>Returns the street that the robot is currently at.</td>
</tr>
<tr>
<td>getDirection</td>
<td>Returns the direction the robot is facing</td>
</tr>
</tbody>
</table>

Robots

- Java: Learning to Program with Robots
  - Based on Rich Pattis’ Karel the Robot
  - Teaches basic Java concepts in the context of graphical world
    - Objects and methods
    - Decomposition and abstraction
- World
  - Cities with streets and avenues
    - Things that can sit at particular intersections, be picked up, and put down
    - Robots that can move around cities and manipulate Things

Constructing a robot

- Robot karel = new Robot(waterloo, 1, 0, Directions.SOUTH);
- name of constructor
- name chosen for object
- parameters: information needed by constructor to create object
- type of object being created
- keyword indicating a new object is being created
- semi-colon to end statement
Calling a method

- Objects
  
  name of object providing a service
  name of service requested
  "dot" connector
  semi-colon to end statement
  no parameters (no information needed)

Decision trees

- If-Then statements
  ```java
  if (logical expression) {
      "true" actions
  }
  ```
- If-Then-Else statements
  ```java
  if (logical expression) {
      "true" actions
  } else (logical expression 2) {
      "false" actions
  }
  ```
- Logical expressions
  - analogous to yes or no questions
  - true or false
- Statements that are true
  - (5 < 7)
  - (100 == 100)
  - (100 != 10)
  - (10 <= 10)
- Statements that are false
  - (-2 > -1)
  - (10 != 10)

A decision tree

- Would you like to read about Einstein?
  - He received the Physics Prize in 1921.
  - Would you like to read about a scientist?
    - Try A. Solzhenitsyn, Literature 1970.
    - Would you prefer a humanitarian?
      - Look up the Peace Prize in 1991.
      - Would you like to read about a scientist?
        - Try the Medicine Prize in 1962.

More Java Syntax

- Assignment statement
  ```java
  variable = expression;
  ```
- Method invocation
  - Also called function or procedure
  - Invoking also called "calling" a function
  - Methods can take arguments
    ```java
    button.setText("This text is an argument");
    init();
    ```
- Variable declaration
  ```java
  VariableType variableName;
  Button choice;
  ```
Java Details

- **Java tutorial** [http://java.sun.com/docs/books/tutorial](http://java.sun.com/docs/books/tutorial)
  - Do “Your First Cup of Java” and create your First Applet
  - Go to “Learning the Java Language” and read “Language Basics”
- **Variable**: an item of data named by an identifier
- **Operators**
  - Arithmetic
  - Relational and conditional
  - Assignment
  - Other
- **Expression**: a series of variables, operators, and method calls that evaluates to a single value

Dealing with numbers

- **Primitive data type**: int
  - Does not require a `new` statement to create
  - Primitive types not classes
  - Must declare
  - Should initialize (Java sets to 0)
  - Other primitive types include: boolean, char, double
- **Operations using integers**
  - +, -, *, /, %
  - Operator Precedence

Some arithmetic details

- Java adheres to traditional order of operations
  - * and / have higher precedence than + and –
    - `int x = 3 + 5 * 6; int y = (3 + 5) * 6;`
  - Parentheses are free, use them liberally
- Arithmetic expressions are evaluated left-to-right in the absence of parentheses
  - `int x = 3 * 4 / 6 * 2; int y = (3*4)/(6*2);`
- There are limits on int and double value, be aware of them.

Types for Numbers

- The type String is not a built-in type, technically it’s a class
- There are many numerical types in Java We’ll use two
  - int, represents integers: [...-3,-2,-1,0,1,2,3,...]
    - Conceptually there are an infinite number of integers, but the range is limited to `[-2^31, 2^31-1]` or `Integer.MIN_VALUE, Integer.MAX_VALUE`
    - Alternatives? Why is range limited?
  - double, represents real numbers like π, √2
    - Not represented exactly, so expressions like 100*0.1 may yield unexpected results
    - Double precision floating point numbers, another type float exists, but it’s a terrible choice (generates poor results)
Dealing with text

- Strings are a class and not a primitive datatype
- Declaration:
  ```java
  String message;
  ```
- String Constants
  ```java
  "Good Morning World!"
  ```
- String Assignment
  ```java
  message = "It's Friday";
  ```

GIGO: program as good as its data?

- In calculations involving floating point numbers it’s easy to generate errors because of accumulated approximations:
  - What is \( 10^{23} + 1 \)?
  - When is \( (x + y) + z \) different from \( x + (y + z) \)?
- The type int is severely constrained on 16-bit computers, e.g., running DOS, largest value is \( 2^{15} - 1 \)
- Even on 32-bit machines, how many seconds in a millennium? \( 60 \times 60 \times 24 \times 365 \times 1000 \), problems?
- On UNIX machines time is measure in seconds since 1970, problems?
- What was Y2K all about?

What arithmetic operations exist?

- Syntax and semantics for arithmetic operations
  - Addition, subtraction: + and -, int and double
    ```java
    23 + 4  x + y  d - 14.0 + 23
    ```
  - Multiplication: *, int and double
    ```java
    23 \times 4  y \times 3.0  d \times 23.1 \times 4
    ```
  - Division: /, different for int and double
    ```java
    21 / 4  21 \div 4.0  x / y
    ```
  - Modulus: %, only for int
    ```java
    21 \mod 4  17 \mod 2  x \mod y
    ```
- Mixed type expressions are converted to “higher” type
- Associativity of operators determines left-to-right behavior
- Use parentheses liberally
  - Without () use operator precedence, * /, % before +,-

Manipulating Strings

- Methods for manipulation
  ```java
  int length()
  ```
  ```java
  int indexOf(String st)
  ```
  ```java
  String substring(int start, int end)
  ```
- Getting String Data from user
  - The TextField class has getText() method
  - Use:
    ```java
    message = mg.getText();
    ```
    - where mg is a TextField and message is a String
Evaluating expressions

- Order of precedence

<table>
<thead>
<tr>
<th>Operators</th>
<th>Associtivity</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>()</td>
<td>left to right</td>
<td>Parentheses</td>
</tr>
<tr>
<td>* / %</td>
<td>left to right</td>
<td>Multiplicative</td>
</tr>
<tr>
<td>+ -</td>
<td>left to right</td>
<td>Additive</td>
</tr>
<tr>
<td>&lt; &lt;= &gt;</td>
<td>left to right</td>
<td>Relationals</td>
</tr>
<tr>
<td>== !=</td>
<td>right to left</td>
<td>Equalities</td>
</tr>
<tr>
<td>=</td>
<td>right to left</td>
<td>Assignment</td>
</tr>
</tbody>
</table>

- Automatic type conversion
  - Values of one type are promoted to another compatible type as part of the computation process.

- You can convert $T_f$ degrees Fahrenheit to $T_c$ degrees Celsius using the formula:
  $$T_c = \left(\frac{5}{9}\right)(T_f - 32)$$

- Given the following expression:
  $$\text{double } T_c = (\text{TF} - 40.0) \times (\frac{5}{9})$$
  If $T_f$ is -40.0 what is $T_c$?
  1. -40.0
  2. 0.0
  3. 40.0
  4. error
  5. unknown

Loops

- If statements need to repeat, then you probably need a loop
- Describe portion of program as:
  - Repeat
  - Continue until
  - For each value from 1 to n
  - For every object of a set, do something
- We have already used iteration by using the buttons
  - How?

Problems

- We want to:
  - Print out all numbers from 0 up to 100 incrementing by 0.5 each time
  - Sum up the numbers from 1 to 100
  - ...

- New Java syntax
  - New object type `TextArea` which is basically a big scrolling textbox
  - `TextArea` is 80 character wide and 20 rows high text box with 20 rows
    ```java
    TextArea tArea = new TextArea(20, 80);
    ```
  - Add characters to the end of the `TextArea` using `append`
    ```java
    tArea.append("Hello\n");
    ```
  - `\n` is called a newline character which moves the next character to the next line
Anatomy of a while loop

- While loops are one way to get rid of repetitive code
- Print out numbers up to 100 by increments of 0.5

```
x ← 0
while (x < 100)
  x ← x + 0.5
  print x
```

Another loop

- Summing the numbers 1 ... 100
  ```
  int sum = 0;
  int k = 0;
  while (k < 100)
    k = k + 1;
    sum = sum + 1;
  }
  ```
- Other Loop designs
  - Count down
  - Stopping and starting at computed values
  - Data dependent loop

Functions/Methods

- Function example: distance from point (x,y) to origin
- Function declaration
  - Name of the function
  - Type of each argument to the function with a descriptive name for each argument
  - The type of value a function returns

```
function distance(x, y)
  return sqrt(x^2 + y^2)
```

Function calling mechanics

- The value of each argument are computed
- The value of each argument is copied into the corresponding formal parameter
- The statements in the function body are evaluated until a return statement appears
- The value of the return expression is evaluated
- The calling program continues, with the returned value substituted in place of the call

```
result = distance(3, 4)
```