More Types

Sep 18

double, boolean
HW1 notes

• Java conventions
  • classes start with initial capital letter    ---- public class MyDemoClass
  • variables start with initial lower case letter    --- int aSingleInteger;
  • methods start with an initial lower case letter --- public static void main.....
  • after that, use CaMeL case

• Use
  • int rather than Integer
  • double rather than Double

• Spaces in file names -- just say NO

• 100% max

• Concatenation and the ++ problem

• ls -a and cat to check what you are submitting

• "package XXX" appears at the top of your program
the 'double' type
floating point numbers

• double base = 55.0;  //64 bits
  1000000010110110000000000000000000000000000000000000000000000000

• Recall
  • int base = 55;  //32 bits
    00000000 00000000 00000000 00110111
public class FunWithDouble {
    public static void main(String[] args) {
        double a = 11.4;
        double b = 16;
        double c = a + b;
        c = 4.2e-5;

        double k = 19 + 0.25;
        double m = k / 3.5;
        String valu = "3.14159";
        double pi = Double.parseDouble(valu);

        double x = 3 * 4;

        double y = 3 / 4;

        int a = 2;
        double b = a + 3.4;

        int mi = 8;
        double n = 7.7;
        int z = mi + n; //
    }
}
Converting to / from double

- String to double
  - double baseDouble = Double.parseDouble(baseString);

- int to double
  - double baseDouble2 = baseInt;  // automatic

- double to int
  - int baseInt = (int)baseDouble;  // NOT automatic
  - MUST "cast"
  - int baseInt = (int)baseDouble;
Casting
converting one type to another

• Often Java will convert types for you
• Sometimes, you need to tell Java exactly what you want
  • this is called "Casting"
• For example, the code at right fails to compile with the message

```
public static void main(String[] args) {
    double aDouble = 2.2;
    int anInt = 4;
    double resultDouble = aDouble * anInt;
    int resultInt = aDouble * anInt;

    System.out.println(aDouble + " * " + anInt + " = " + resultDouble);
    System.out.println(aDouble + " * " + anInt + " = " + resultInt);
}
```

Casting1.java:6: error: incompatible types: possible lossy conversion from double to int
    int resultInt = aDouble * anInt;

• To fix, cast
  • change line to (int)(aDouble*anInt);
• When you do this, what does the program print?

```
int resultIntB = (int) aDouble * anInt;
System.out.println(aDouble + " * " + anInt + " = " + resultIntB);
```
• The formula for "wind chill" is
  \[\text{windChill} = 35.74 + 0.6215 \times \text{Temperature} + (0.4274 \times \text{Temp} - 35.75) \times \text{windSpeed}^{0.16}\]

• To compute a number raised to a "weird" power use Math.pow(a,b). e.g.
  ```java
  double number = 16;
  double power = 0.25;
  double res = Math.pow(number, power);
  System.out.println(res);
  ```

• Recall to get a double from the command line:
  ```java
  double argDouble = Double.parseDouble(args[0]);
  ```

• Write a program to compute (and show) the wind chill temperature given the current temperature and wind speed as command line arguments
Yet more Types, Boolean

• Just true or false.
  ```java
  boolean trueValue = true;
  boolean falseFalse = false;
  ```

• if that was all, booleans would be boring
Boolean Comparison Operators

• They return true or false
• > (also <, <=, >=)
  • Not <=
• ==
• !=
public class AnalyzeNumber {
    public static void main(String[] args) {
        int num = Integer.parseInt(args[0]);
        boolean isPositive = num > 0;
        boolean isZero = num == 0;
        boolean isNotZero = num != 0;
        boolean isEven = num % 2 == 0;
    }
}
Combining booleans
and and or and not

- Boolean values can be combined!!!
  - &&
  - ||
  - !

Boolean "Truth Table"

<table>
<thead>
<tr>
<th></th>
<th>and</th>
<th>or</th>
<th>not</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>TRUE</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>FALSE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
• Doing things conditionally
• if (true) {
    do this
}