

More on Variables

The Nitty Gritty

Overview

- Types
 - Fundamental, e.g. int, double, boolean, char
 - Object, e.g. Robot, World, RightTurner, CleverRobot
- Cannot mix types
 - (exception: double can have int value)
- Increment/decrement, e.g. `i++ j--`
- Names, Rules and conventions
- Constant Variables

Data Types and Expressions

- Different kinds of information
- Each kind is a "type"
- Java has many data types
 - Each type has a name
 - Each type has a set of *literals*
 - Each type has a set of operations
- Data types in Java can be *fundamental types* or *object types*

Also called
primitive
or simple
types

Fundamental Types in Java

- **int**
 - represents integers
 - literals like 3 and 132
 - operators like +, -, *, /, %
- **double**
 - represents real numbers
 - literals such as 3.0 or 3.14159 and 2.997925e8
- Representation
 - The integer 3 and the double 3.0 are represented differently in the computer

More Fundamental Types in Java

- **boolean**
 - Represents true and false
- **char**
 - Represents individual characters that are typed at the keyboard

Fundamental Types

Type	Range		holds
int	2billion	-2billion	whole numbers
double			floating point numbers
char			printable items, e.g. letters.
boolean	true	false	a condition.

Less common fundamental types

- The other four fundamental types are variations on an **int** and a **double**; **byte**, **short** and **long** are variations of **int**, **float** is a variation of **double**. These types are rarely used.

Stick to Your Own Kind

- Java is very strict about data types in assignment statements
- Java only puts into a variable a value of the appropriate type
- Expressions having an integer value can go into an integer variable, etc.

```
int i;  
double x = 3.0;  
i = 10.3 * x; ← Illegal statement
```

Exceptions

- You *can* assign an integer expression to a double variable
 - The integer is converted to a double without loss of information, so it is done automatically
- You can do this:

```
int i = 3;  
double x;  
x = 10 + i;
```

Converting between types

- The *cast* operation lets you explicitly convert one type to another
- You can do this:

```
int i;  
double x = 3.0;  
i = (int) (10.3 * x);
```

- The cast (*int*) takes the expression that follows it and converts it into an integer by removing any fractional part; *i* is assigned the integer value 30

Increment/Decrement

- There are two special cases of the assignment statement, incrementing and decrementing

cent++;

is equivalent to

cent = cent + 1;

cent--;

is equivalent to

cent = cent - 1;

Statements

- The temperature program is a sequence of statements, each statement either assigning a value to a variable, or an executable statement that prints output

```
System.out.print("Enter temperature (deg C): ");  
temperature = Console.readInt();
```

- There are of course other kinds of statements, conditionals (*if*, *if/else*), and iterative statements (e.g. *while*)

Aside on Names

- We have seen three ways of defining things: Class definition, method definition and variable declaration.
- Name Rules
 - May contain any sequence of letters, underscores and digits as long as it does not start with a digit.
 - Exception: it may not be a Java keyword
 - Names may not contain spaces

Examples

```
public class CleverRobot
{
}
```

Class definition

```
void turnRight()
{
}
```

Method definition

```
int temperature;
```

Variable declaration

Style Guide for Names

- Names should be meaningful, e.g. CleverRobot, not Y3x
- Names of classes should start with a capital letter, all other names should start with a lowercase letter.
- If a name consists of more than one word, then all words other than the first should start with a capital letter. All other letters should be lowercase: E.g. `matchOfTheDay`

Advantages of Style Guide

- When you see a name, you know
 - if it starts with a capital it's a class
 - otherwise
 - if it ends in parentheses, it's a method
 - otherwise it's a variable
- e.g. `CleverRobot, temperature, jim, move(), String, print()`

Constant Variables!

- We can define a variable (a "symbolic constant") whose value will never change, by writing
`final int RETIREMENTAGE = 70;`
- This carries out assignment at the same time as the definition (as we can do with any variable)
- Now RETIREMENTAGE cannot be legally changed in the program

Summary

- Types
 - Fundamental, e.g. int, double, boolean, char
 - Object, e.g. Robot, World, RightTurner, CleverRobot
- Cannot mix types
 - (exception: double can have int value)
- Increment/decrement, e.g. `i++ j--`
- Names, Rules and conventions
- Constant Variables

Questions

