CSE 142

Declaration, Assignment, & Expressions

The Fine Print

Overview

- Review:
  - Names, bindings, declarations, initialization & assignment
- Today
  - Details about expression evaluation and assignment
  - Conversions: mixed-mode arithmetic, numbers, and strings
- Reading
  - Dugan notes: part of ch. 7
  - Niño & Hosch: sec. 5.2

Names can be Declared and Initialized…

- A name is *created* using a *declaration*

```
double balance;
```

- A name is *initialized* using an *assignment* statement

```
balance = 0.0;
this.balance = 0.0;
```

- We can do both in one step for a local variable

```
double balance = 0.0;
```

- We have to separate them for instance variables
- Parameter lists are declarations of the parameter names

```
public void transferTo(double amount, BankAccount destination) { … }
```

… and Rebound

- A name can be *rebound* to a different value using an assignment.

```
this.balance = this.balance + amountToDeposit;
```

- The name now refers to the new value
- Assignments are statements, and can appear anywhere a statement is allowed. (Just like if statements, declarations with and without initialization, etc.)

Assignment Patterns

- Pattern of an assignment to a variable in scope:

```
<name> = <expression> ;
```

- Pattern of an assignment to an object's instance variable:

```
<object name> . <instance variable name> = <expression> ;
```

- Execution of an Assignment
  1) Evaluate <expression>
  2) Bind value to name
- This ordering means that statements like

```
count = count + 1;
```

have a well-defined meaning

How Expressions are Evaluated

kind of expression	examples	value
literal	9 'b' "Bill"	the literal value
creation of a new object	new House( ) new Rectangle(...)	the newly created object
name of an object	this.frame myMoney	the object the name refers to

message send to an object	this.frame.getX( )
	myMoney.getBalance( )
		the value the method returns

result of an operator	box.getX( ) + 173 acct.getBalance( )<100.0	depends how the operator works

Value-returning Methods

- The result of a method call can appear in an expression

```
fantasyBalance = myMoney.getBalance( ) * 1000.0;
```

- The object must include a method with an appropriate return type

```
/** Access account balance
 * @return current balance of this account */
public double getBalance( ) {
    return this.balance
}
```

- Execution of the return statement:
  - Designates the expression value returned by the method, and
  - Immediately stops execution of the method & returns that value

Kinds of Numbers

- Java provides two main numeric types
  - Integers (int) – exact whole numbers; finite range (approx. ±2147483647)
  - Floating-point (double) – scientific notation; finite precision (about 14 decimal digits), but much wider range ($10^{\pm 308}$) [Dugan notes call these "rational numbers"]
- We sometimes have one kind of number and need to use it where the other kind is expected
  - Example: we have a double, but need an int for a graph
  - Example: we have an int, but want to call a method with a parameter of type double

Numeric Conversions – Casts

- To convert a double *d* into an int, use a *cast*: (int)*d*
  - Fractional part of the number is discarded

```
double totalRainfall = 123.45;
int rectangleHeight = (int)totalRainfall;
```

  - (int) is a kind of unary operator, with high precedence, so need parentheses for complicated double expressions.

```
int smallRectangle = (int)(totalRainfall / 20.0);
```

- Don't need an expression to convert an int into a double; Java will do it automatically
  - Idea: int->double retains the original value, adding a ".0". double->int might lose information; programmer is required to show that was intended by using a cast

Mixed-Mode Arithmetic

- If ints and doubles are combined in an expression, the int values are treated as doubles

```
int numberOfWidgets = 17;
double pricePerWidget = 1.42;
double totalPrice = numberOfWidgets * pricePerWidget;
```

  - (does not change the actual value stored in int variables)
- Usually works as expected, but beware:

```
double pricePerBox = 12.95;
double priceOfOrder = pricePerBox * 2 / 3;
```

vs

```
double priceOfOrder = 2 / 3 * pricePerBox;
```

Numbers to Strings

- Recall that "+" can be used to concatenate two strings

```
String greeting = "hello " + "there"
```

- If "+" is used to concatenate a string and a number, the number is converted to the corresponding string of characters

```
double height = 45.6;
String howBigIsIt = "It is " + height + " cubits tall";
```

- Can concatenate a number to a null string to get a string representation of the number

```
String classNumber = "" + 142;
```