CSE 142

Types and Messages

Overview

- Topics
  - Types
  - Type compatibility
  - Syntax and semantic errors
  - Messages and interfaces; parameter lists
- Reading
  - Dugan notes: middle of ch. 5

Types

- Some expressions:

```
"hello"
aSquare
aSquare.getColor( )
aSquare.getX( ) * 24
```

- What kind of thing does each expression evaluate to?
- Java calls "kinds of things" *types*.
- When we create a new name, we must always specify a type:

```
String greeting = "hello";
```


Type Mismatches

- What's wrong with this sentence: "Her age is green."
- Java is extremely picky about these kinds of errors, which we call *type mismatches*. For example:

```
int myAge = "green";
String greeting = 23;
```

- Why do you think that Java is picky about this kind of thing?

Kinds of Errors

- Some odd sentences:
  - *They has went, to Tibet to "climbed mountain.)*
  - *Curious green ideas sleep furiously.*
  - *To get a good grade, you should skip lecture.*
- Each of these is wrong in some way:
  - The first has *syntax errors* (grammar and punctuation mistakes).
  - The second has *semantic errors* (misuse of types): we don't think of ideas has having colors, or sleep being furious, etc.
  - The third is a *broken algorithm*: it doesn't accomplish the task we intended.
- The Java interpreter catches syntax and semantic errors automatically.
- Broken algorithms must be found by the programmer through thinking or testing.

Syntax & Semantic Errors

- What is wrong with each of these statements? Syntax or semantics or algorithm?

```
int x is 7;

String myName = 7;

aSquare.moveBy("hello", 40);

int velocity = distance / time

String greeting = "hello" 56;

int area = length * "width";

double blackHoleGravity = mass / 0.0;
```

Message Parameters

- Many messages require some information:

```
aRectangle.moveBy(10, 20);
```

- We call the items we send with a message *parameters* or *arguments*.
- Each parameter may be any expression, however complicated, as long as the type and number of parameters matches what the message expects:

```
aRectangle.moveBy(10 + length * 2, aRectangle.getY( ) + 5);
aRectangle.moveBy("hello", 20);
aRectangle.moveBy(30);
```


Message Interface

- How do we know the right way to send a message? How do we know what messages an object can respond to?
- To really know (and how Java knows): look at the object's *interface*.
- An interface defines: message name, number, type of parameters, and type of the resulting value. It often also has commentary about the behavior of the object when it receives the message.
- *JavaDoc* is a tool for generating web pages from Java code describing their interfaces. See course web for documentation on Rectangle, etc.

Message Interface: Examples

- An excerpt from the moveTo message for Rectangle:

moveTo
public void moveTo(int x,
int y)
Move the upper-left corner of the rectangle to the given coordinates.
Parameters:
x - new X coordinate.
x - new Y coordinate.

  - What does void mean?
- An excerpt from the length message for String (java.lang.String):

length
public int length()
Returns the length of this string.
Returns:
the length of the sequence of characters represented by this object.

Compound Names

- We've seen lots of names that are single words, e.g.:
  x, length, aSquare
- Sometimes we'll have to use *compound names*, which are words linked by dots, e.g.:

```
Color.green
Math.sqrt(3.4)
System.out.println("hello, there!");
java.lang.String
```

- Compound names can only be used in special cases.