CSE 115/503

January 25-29, 2010

Announcements (1/25)

- If you have not picked up a syllabus, please do so
- Sign and return form on last page of syllabus – will be part of Lab 1 assignment.
- Recitation Change Forms
- Lab 1 in recitation this week

Announcements (1/27 & 1/29)

- Account issues
- Web-CAT issues
- Sign and return form on last page of syllabus – will be part of Lab 1 assignment.
- Lab 1 in recitation this week – due Sunday, January 31st
- Exam 1 is Friday, February 5th – review in class on Wednesday, February 3rd

Objects

I- Properties

Features that describe an object

II- Capabilities

Actions that an object can perform

Expressions

Pieces of code that when evaluated produce a result

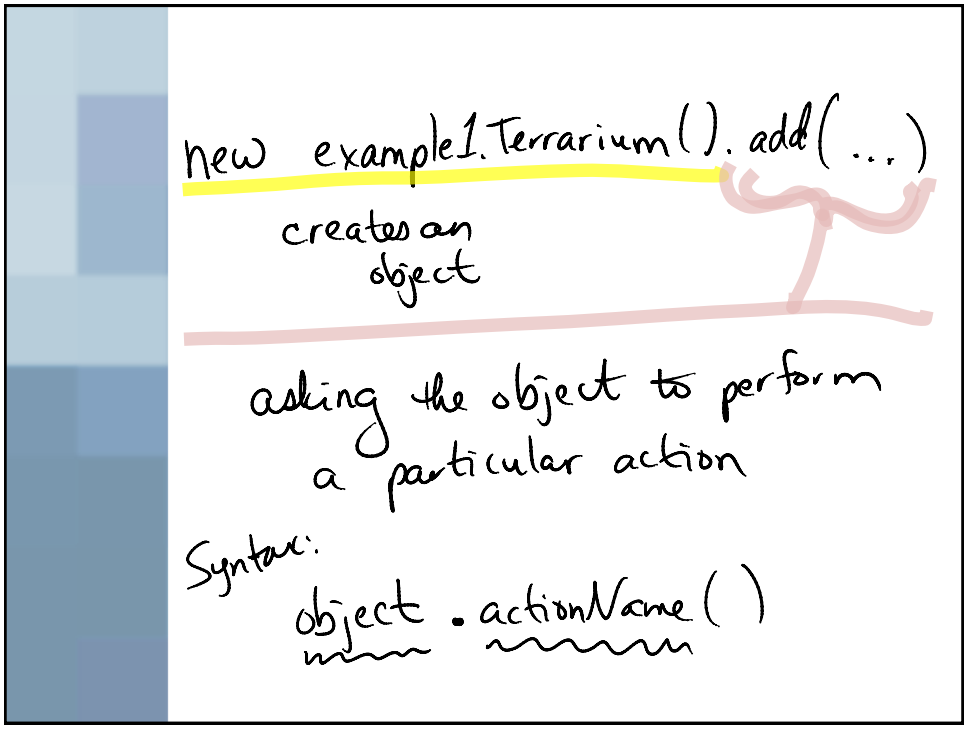
Expression that creates an object

new Name ()

new
↑
keyword

Name
↑
name of object we are creating

()



Creating Objects

- A look inside the machine

(part of) memory

107	
108	
109	
110	
111	
112	
113	
114	
115	

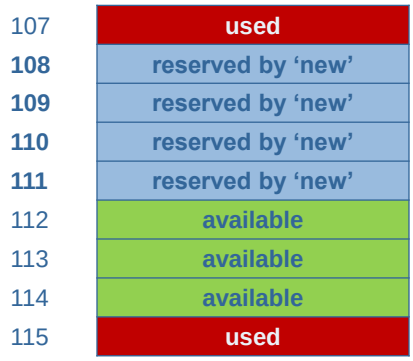
evaluating a ‘new’ expression

When evaluating an expression like ‘new example1.Terrarium()’, the operator ‘new’ first determines the size of the object to be created (let us say it is four bytes for the sake of this example)

107	used
108	available
109	available
110	available
111	available
112	available
113	available
114	available
115	used

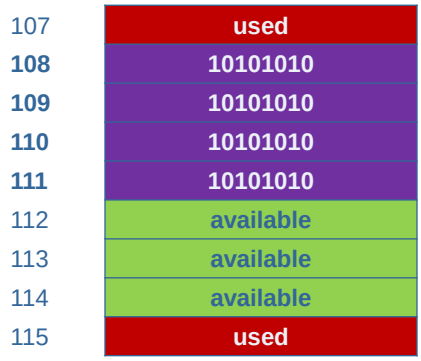
evaluating a ‘new’ expression

Next, new must secure a contiguous block of memory four bytes large, to store the representation of the object.



evaluating a ‘new’ expression

Bit strings representing the object are written into the reserved memory locations.



evaluating a 'new' expression

The starting address of the block of memory holding the object's representation is the value of the 'new' expression. This address is called a 'reference'.

107	used
108	10101010
109	10101010
110	10101010
111	10101010
112	available
113	available
114	available
115	used

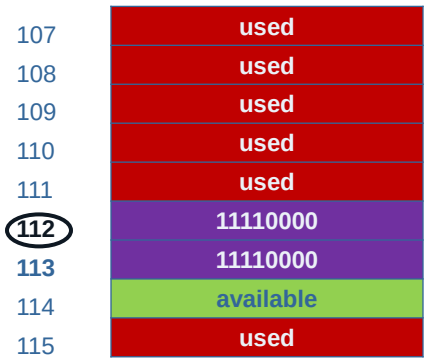
evaluating a 'new' expression

A similar thing happens when we evaluate another 'new' expression like 'new example1.Ant()'.

107	used
108	used
109	used
110	used
111	used
112	available
113	available
114	available
115	used

evaluating a 'new' expression

Supposing that an example1.Ant object occupies two bytes of memory, new reserves a contiguous block of two bytes, writes bit strings representing the object to those memory locations, and the starting address of this block of memory is the value of the 'new' expression.



DrJava's response

When we evaluate these 'new' expressions in DrJava, what is the response we get?

> new example1.Terrarium()

example1.Terrarium[frame0,0,0,608x434,layout=java.awt.BorderLayout,title=,resizable,normal,defaultCloseOperation=EXIT_ON_CLOSE,rootPane=javax.swing.JRootPane[,4,30,600x400,layout=javax.swing.JRootPane$RootLayout,alignmentX=0.0,alignmentY=0.0,border=,flags=16777673,maximumSize=,minimumSize=,preferredSize=],rootPaneCheckingEnabled=true]

DrJava's response

After DrJava evaluates the expression, it must print the value. The way Java works when a reference is printed is that a textual representation of the object it refers to is produced (as defined by the object itself)

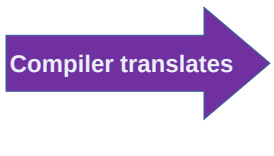
Where do objects come from? (The "birds and bees" lecture)

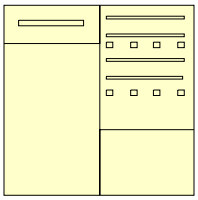
- We've seen how to create an object.
- But where does the object come from?
- How does DrJava know what an example1.Terrarium() object is?

Writing Java Code

- Programmer writes Java code (in an editor, or at the DrJava prompt)
- It is compiled (translated) into a form the computer will understand (by the compiler)

Objects exist only at runtime





Objects do not exist while the programmer writes the program, except in their minds. The programmer writes the code to create the object (new…)

Objects exist only at runtime

Whoa, whoa, wait a minute. You mean to tell me that as object-oriented programmers, we don't write objects?

- That's right – we write class definitions.
- Objects are instances of classes.
- Classes are instantiated only at runtime.

The moral of our story

- So, we will spend a great deal of time writing class definitions and only a small amount of time writing the code to create objects.
- But, at run time, it is the objects that actually do the work – the work we've defined them to do when we wrote the class definitions.

```
new example1.Terrarium()

new example1.Ant()
```

```
new example1.Terrarium().add(new example1.
                             Ant());
```

Values of expressions are lost to us if:

- not used right away
- ~~or~~ remembered (stored somehow)

A variable is:
(at its most basic)

- a storage location in memory
- for example, location 120:

116	
117	
118	
119	
120	*space for a variable*
121	
122	
123	
124	

Variables

- name (given by programmer; Java)
- location (in memory)
- type (representation scheme; size)
- value (contents of the memory location)
- ~~scope~~
- lifetime) "for later"

Why type?

- Tell us how to interpret the bits that are stored

Why name?

Don't know memory address until run-time

Variable declarations

Type identifier ;

- Java requires that we declare variables before we store things in them.

Identifiers (names of things in programs)

- Rules: compiler enforced
- Style: community enforced

Rules

1. identifiers can only contain letters, digits, & underscores
2. identifiers can only begin with a letter or underscore
3. identifiers can not be keywords

Style

- depends on what you're naming
  - package
  - class
  - variable
  - method

variable (local variable)

- start w/ lowercase letter
- letters of subsequent words are ~~capitalized~~ upper case.

eg) myFavoriteArt

Assignment statement

variable = expression;

1-expression is evaluated
2-value is stored in the variable

The following slide…

Shows a memory diagram of executing the following at the DrJava interactions pane:

```
example1.Terrarium terrarium;
terrarium = new example1.Terrarium();
example1.Ant ant;
ant = new example1.Ant();
example1.Ant ant2 = new example1.Ant();
```