Objects and Classes

Object Oriented Programming
2016375 - 5

Camilo López

Outline

- What is an Object?
- Abstraction and Modeling
- What is a Class?
- Declaring a Class – Java Style
- Encapsulation
- User-Defined Types and Reference Variables
- Garbage Collection

What is an Object?

(1) something material that may be perceived by the senses; (2) something mental or physical toward which thought, feeling, or action is directed.

Merriam-Webster's Collegiate Dictionary

Physical Objects

- Students
- Professors
- Classrooms
- Buildings

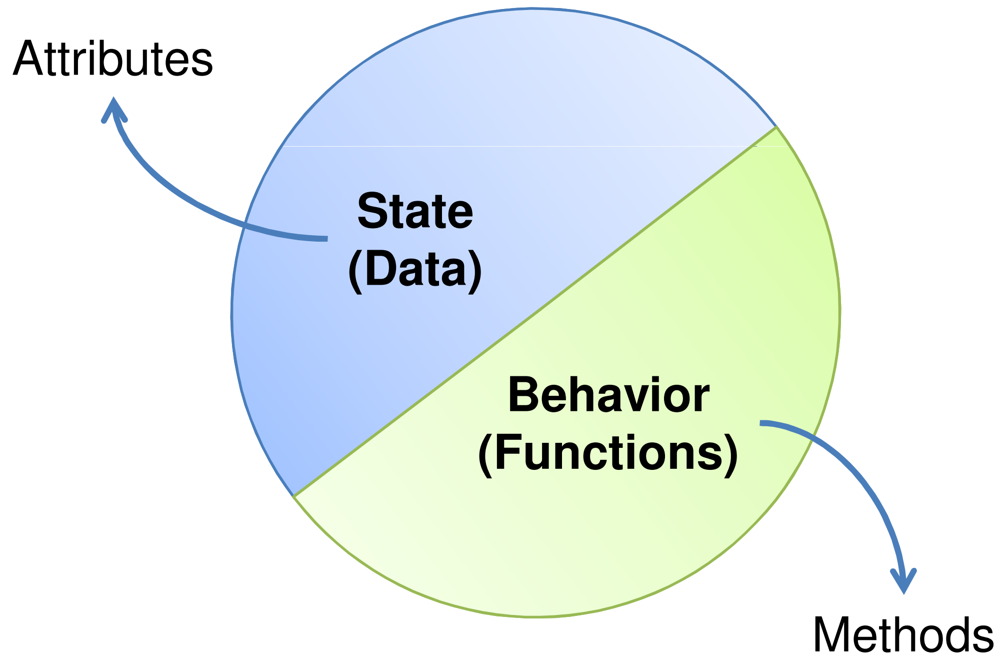
Conceptual Objects

- Courses
- Departments
- Degrees
- Transcripts

Student Registration System (SRS)

What is an Object?

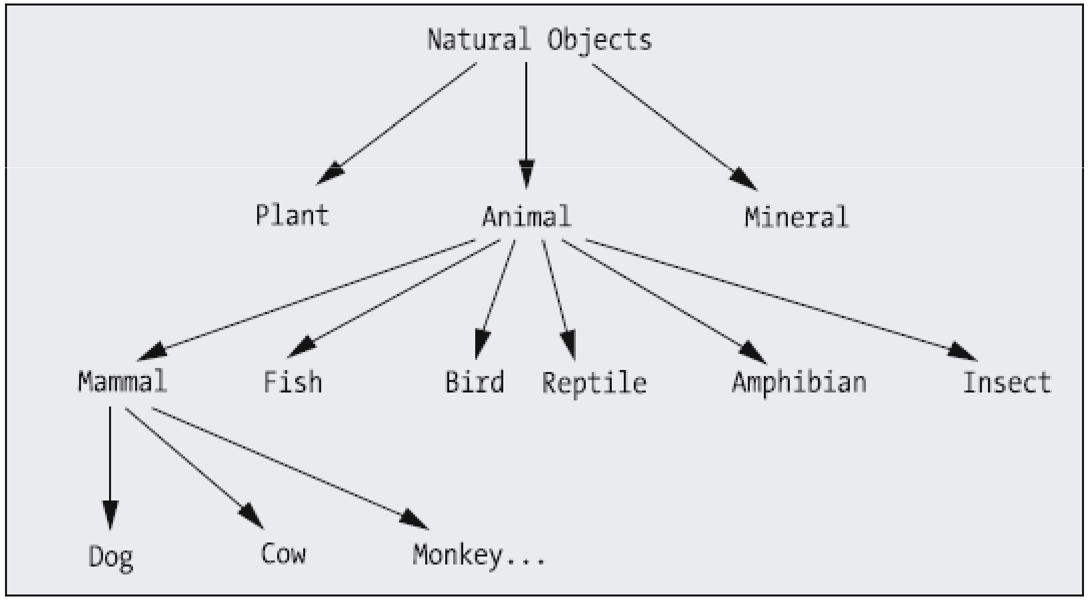
A (software) **object** is a software module that bundles together **state (data)** and **behavior (functions)**



represents an **abstraction** of a real-world object.

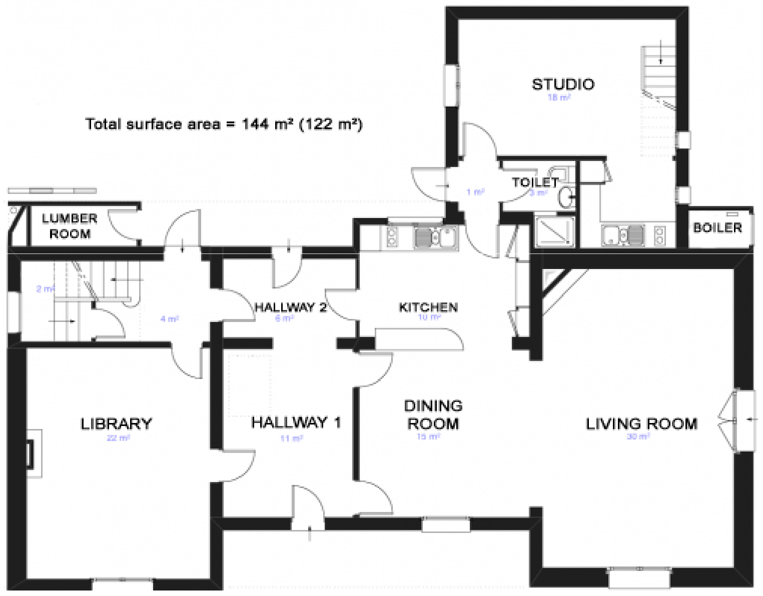
Abstraction and Modeling

- Simplification Through Abstraction
- Generalization Through Abstraction



- Organizing Abstractions into Classification Hierarchies

What is a Class?



What's in the blueprint?

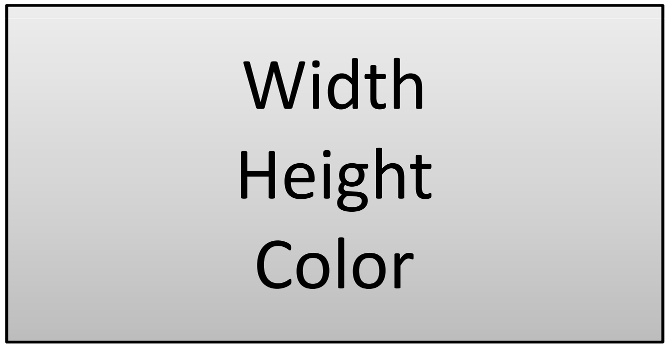
$\downarrow$

Attributes and Behavior

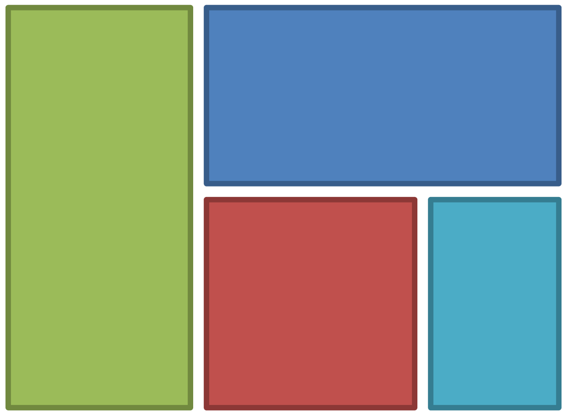
What is a Class?

The blueprint

Objects – Instances of the class



Class Rectangle



Instantiation Is the process by which an object is created in memory at run time based upon a class definition.

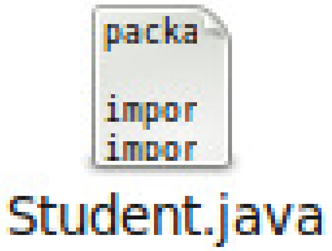
Student Class

Proposed Attributes

Attribute	Type
name	String
studentId	String
birthDate	Date
address	String
major	String
gpa	double
advisor	???
courseLoad	???
transcript	???

Declaring a Class – Java Style

```
public class Student {
    String name;
    String studentId;
    Date birthDate;
    String address;
    String major;
    double gpa;
    // type? advisor – we'll declare this attribute in earnest later!
    // type? courseLoad - ditto
    // type? transcript – ditto

    // ... method declarations
}
```



Allocate a prescribed amount of memory within the JVM to house the attributes of a new object.

Declaring a Class – Java Style

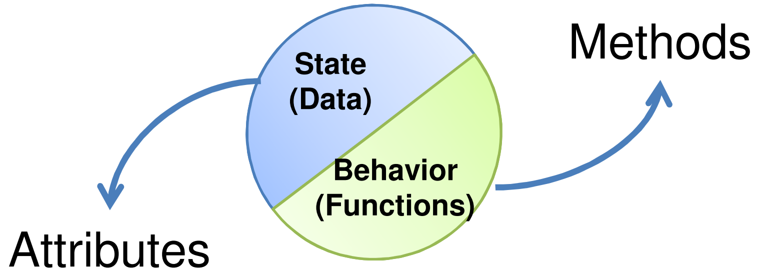
```
public class Student {
    String name;
    String studentId;
    Date birthDate;
    String address;
    String major;
    double gpa;
    // type? advisor – we'll declare this attribute in earnest later!
    // type? courseLoad - ditto
    // type? transcript – ditto

    // ... method declarations
}
```

Associate a certain set of behaviors with that object.

Encapsulation

- the mechanism that **bundles together** the **state and behavior** of an object into a single logical unit.

- Everything that we need to know about a given object is, in theory, contained within the boundaries of a Student object, either
  - Directly, as an attribute of that object or
  - Indirectly, as a method that can answer a question or make a determination about the object's state.



User-Defined Types and Reference Variables

```
int x;
System.out.println(x + 5);
```

The type and a symbolic name

User-Defined Types and Reference Variables

```
int x;
System.out.println(x + 5);

Student y;
```

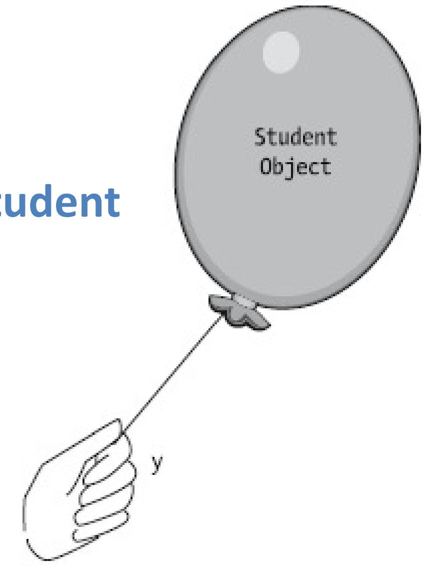
The type and a symbolic name

y is not an Object in memory yet. It’s a reference variable, which has the potential to refer to a Student object

User-Defined Types and Reference Variables

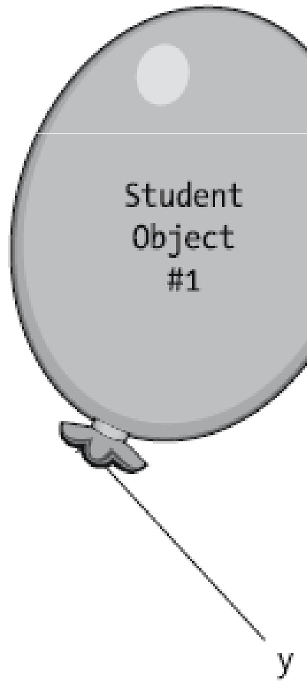
```
int x;
System.out.println(x + 5);

Student y;
y = new Student();
//…
System.out.println(y.name)
```

the special Java keyword, new is used to allocate a new Student object within the JVM’s memory at run time



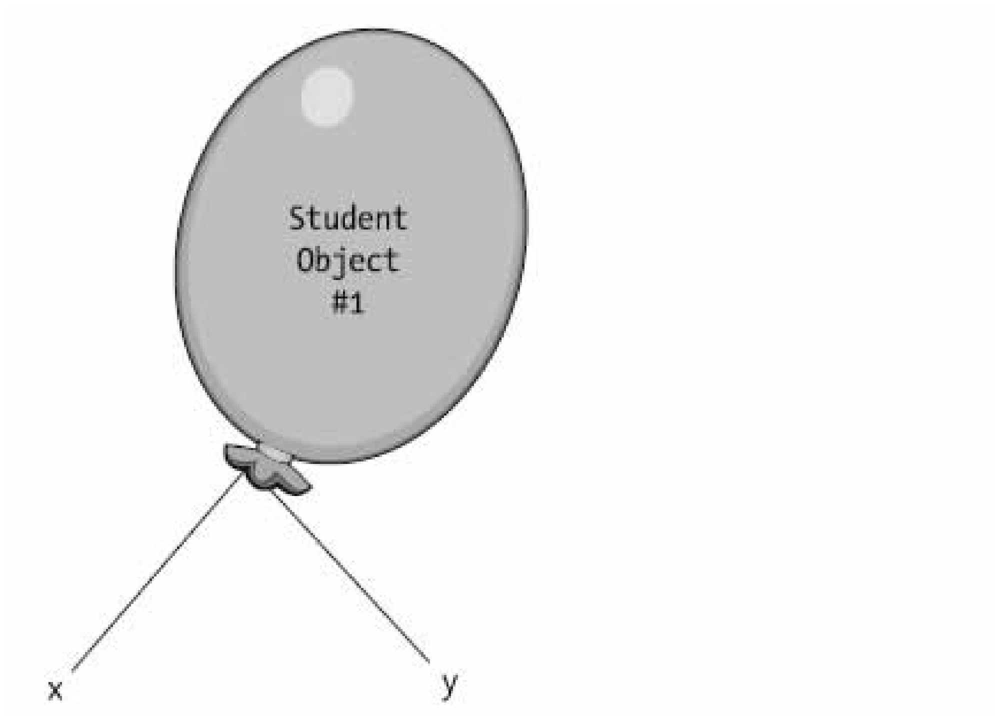
User-Defined Types and Reference Variables

```
Student y = new Student();
```



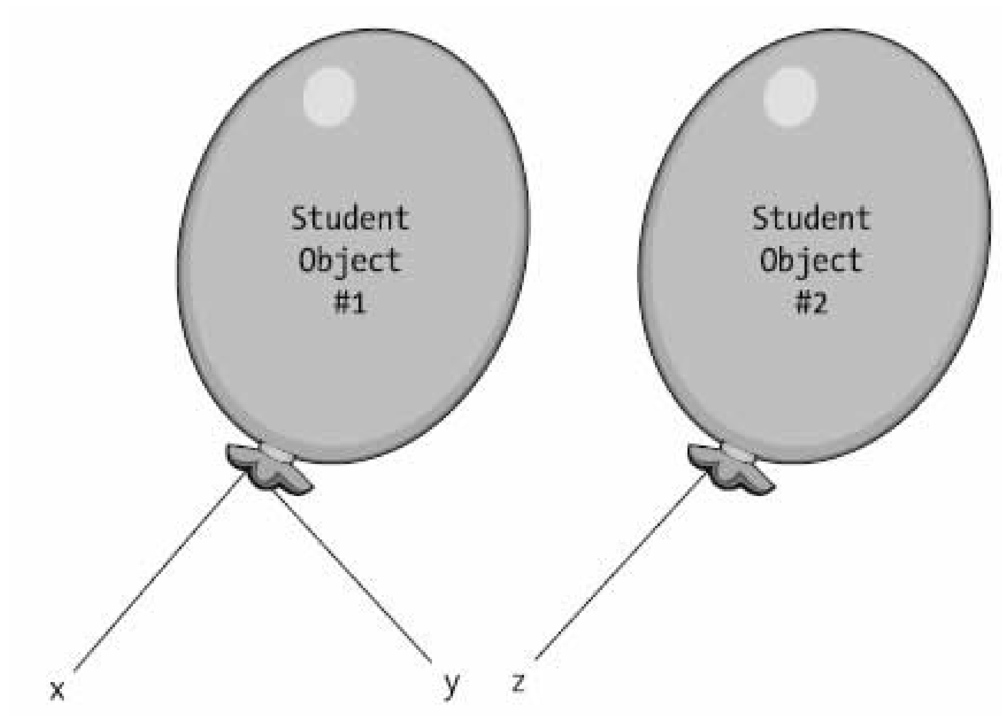
User-Defined Types and Reference Variables

```
Student x;
x = y;
```



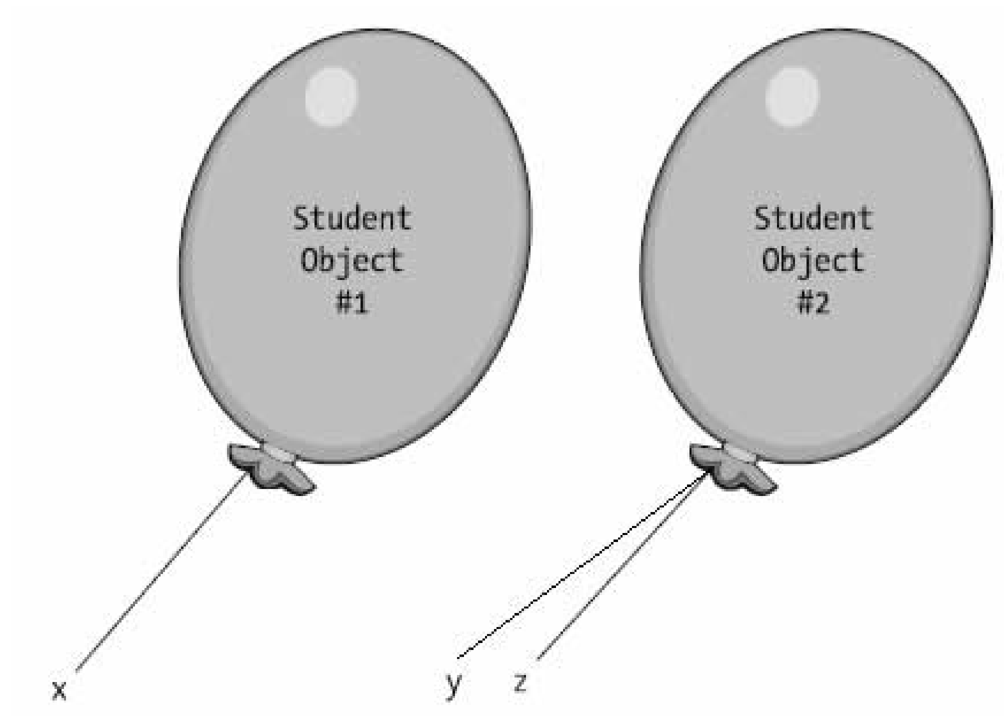
User-Defined Types and Reference Variables

```
Student z = new Student();
```



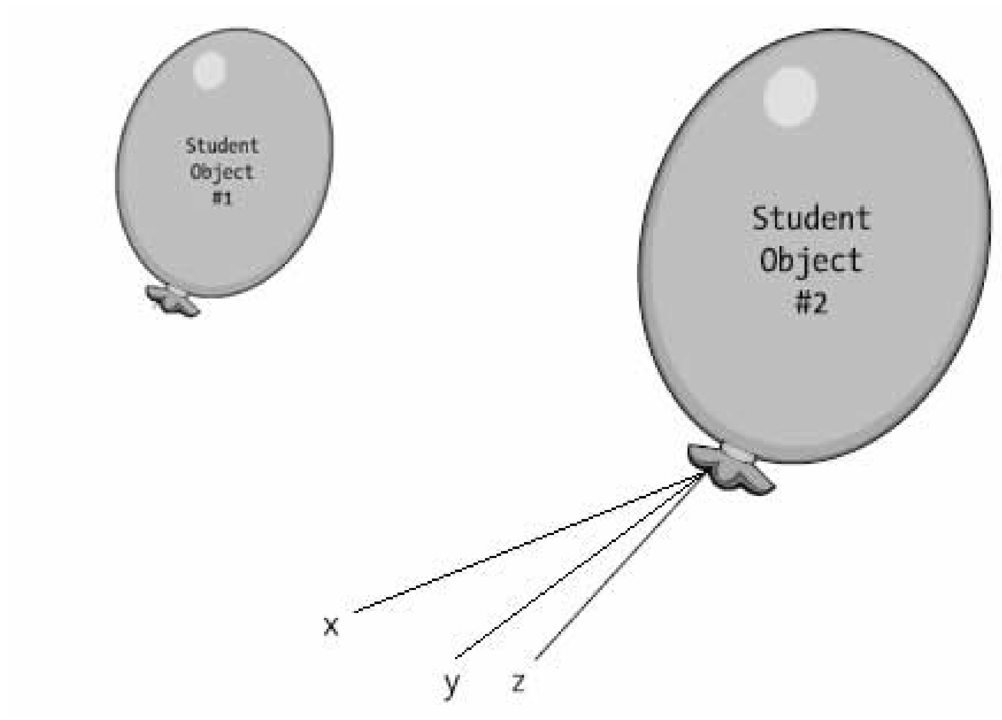
User-Defined Types and Reference Variables

```
y = z;
```



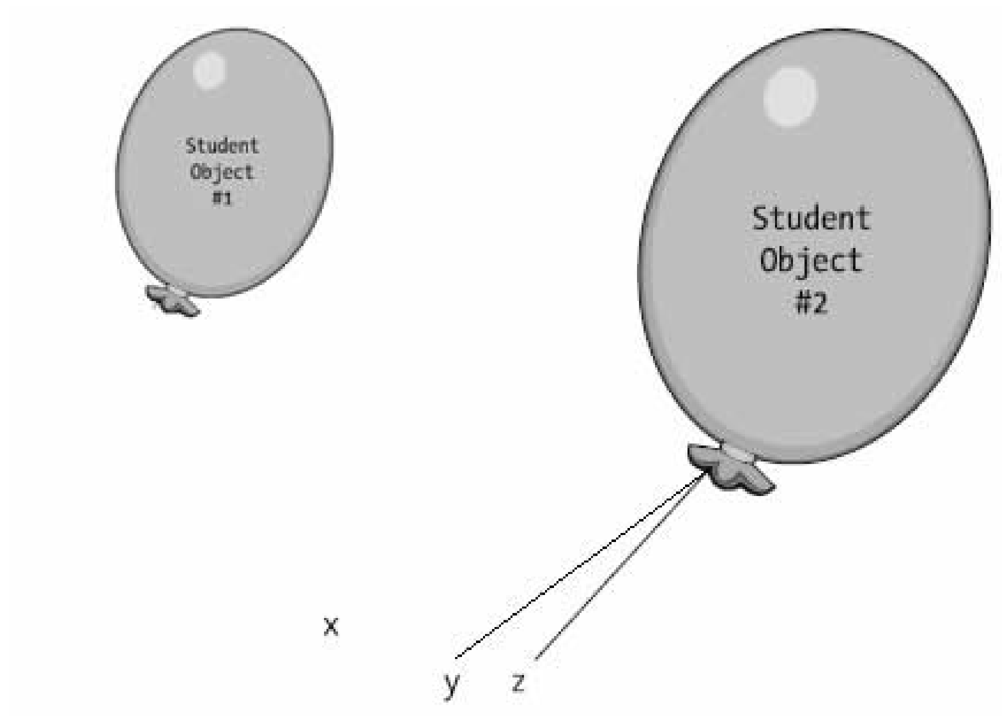
User-Defined Types and Reference Variables

```
x = z;
```



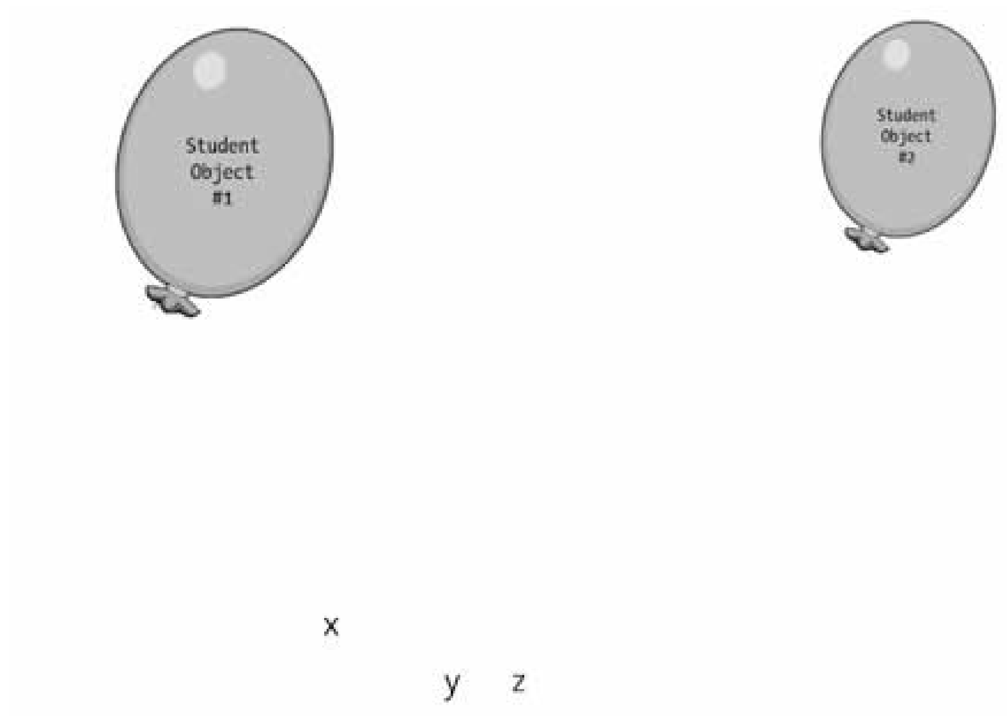
User-Defined Types and Reference Variables

```
x = null;
```



User-Defined Types and Reference Variables

```
y = null;
z = null;
```



Garbage Collection

The JVM periodically performs **garbage collection**, a process that automatically reclaims the memory of “lost” objects for us while an application is executing.

- no remaining active references to an object $\rightarrow$ Candidate for GC
- Garbage collection occurs whenever the JVM determines that the application is getting low on free memory, or when the JVM is otherwise idle.

- there is a way to explicitly request garbage collection to occur in Java via the following statement:

```
Runtime.getRuntime().gc();
```

Objects As Attributes

Attribute	Type
name	String
studentId	String
birthDate	Date
address	String
major	String
gpa	double
advisor	Professor
courseLoad	???
transcript	???

References

- J. Barker, *Beginning Java Objects: From Concepts To Code, Second Edition*, Apress, 2005.
- H.M. Deitel and P.J. Deitel, Java How to Program: Early Objects Version, Prentice Hall, 2009.
- Code Conventions for the Java Programming Language, available at http://java.sun.com/docs/codeconv/CodeConventions.pdf