OOP, Classes and Objects

Christian Rodríguez Bustos

Edited by Juan Mendivelso

Object Oriented Programming

Agenda

1. How our brain manage knowledge
2. Object Oriented Approach
3. Objects in Java
4. Instantiating Objects: A Closer Look
5. Exercise

1. How our brain manage knowledge

1.1 Abstraction
1.2 Abstraction hierarchy
1.3 Abstraction and software development

How our brain manage knowledge

How our brain manage knowledge

What do you remember?

1.1 Abstraction

How our brain manage knowledge

Our brains naturally **simplify** the details of all that we observe.

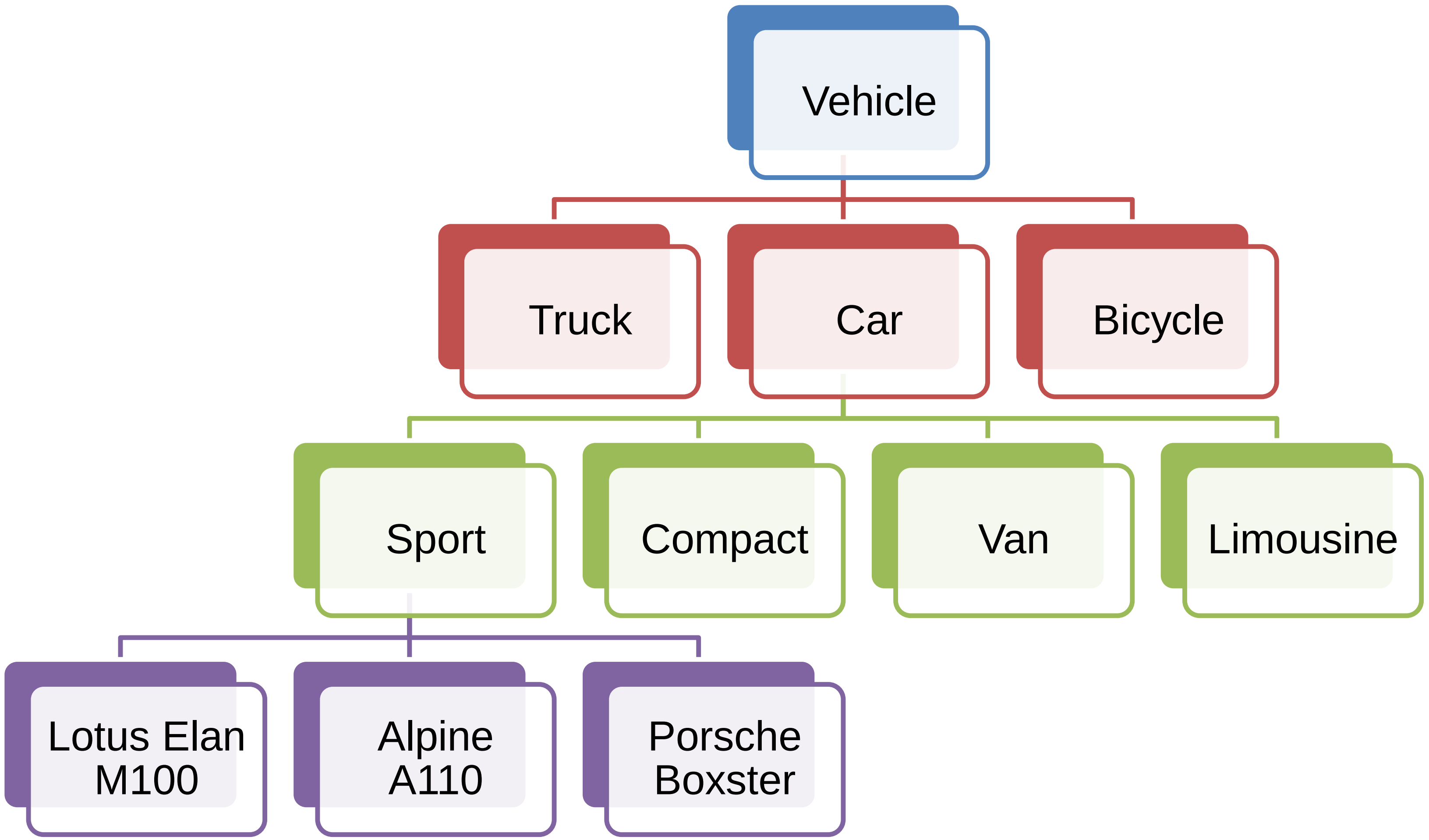
Details are manageable through a process known as **abstraction**.

Abstraction

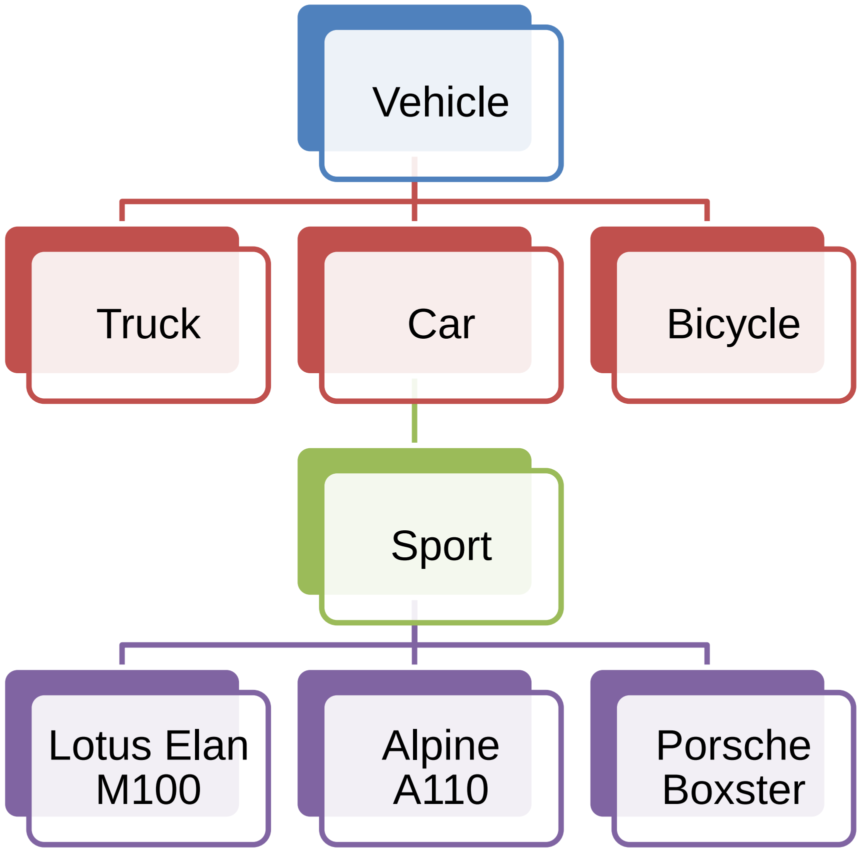
Process that involves **recognizing and focusing** on the **important characteristics of a situation or object**, and filtering out or ignoring all of the **unessential details**.

1.2 Abstraction Hierarchy

Simple abstraction hierarchy



Simple abstraction hierarchy



Focusing on a small subset of the hierarchy is less overwhelming.

Sport car rules

- Small
- Two seat
- Luxury
- High speed

Simple abstraction hierarchy

Correct classification

1.3 Abstraction and Software Development

Abstraction and software development

Developing an abstraction of the problem is a necessary first step of all software development.

2. Object Oriented Approach

2.1 Objects & Classes

2.2 State / Data / Attributes

2.3 Behavior / Operations / Methods

What is OOP?

The logical approach used in software engineering that describes how a programming language is implemented.

It is a **programming paradigm** where developers think of a program as a **collection of interacting objects**

What is a 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

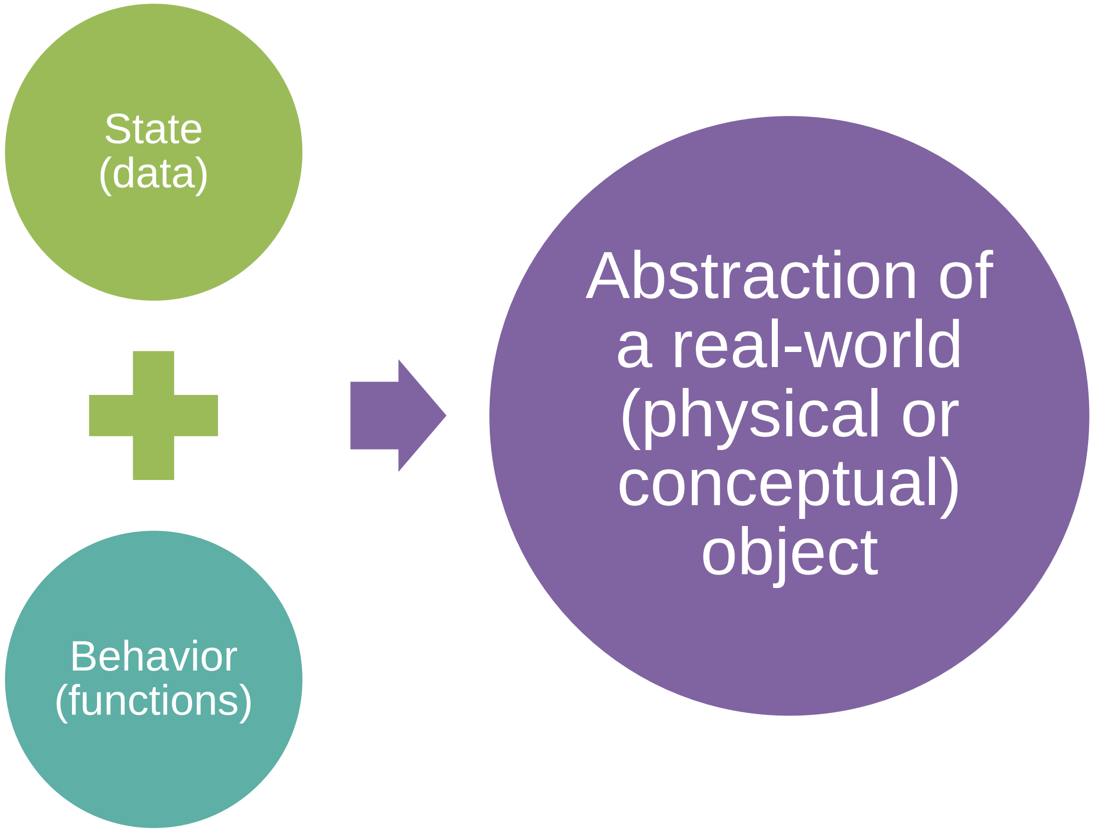
Physical objects

- Person
- Student
- Professor

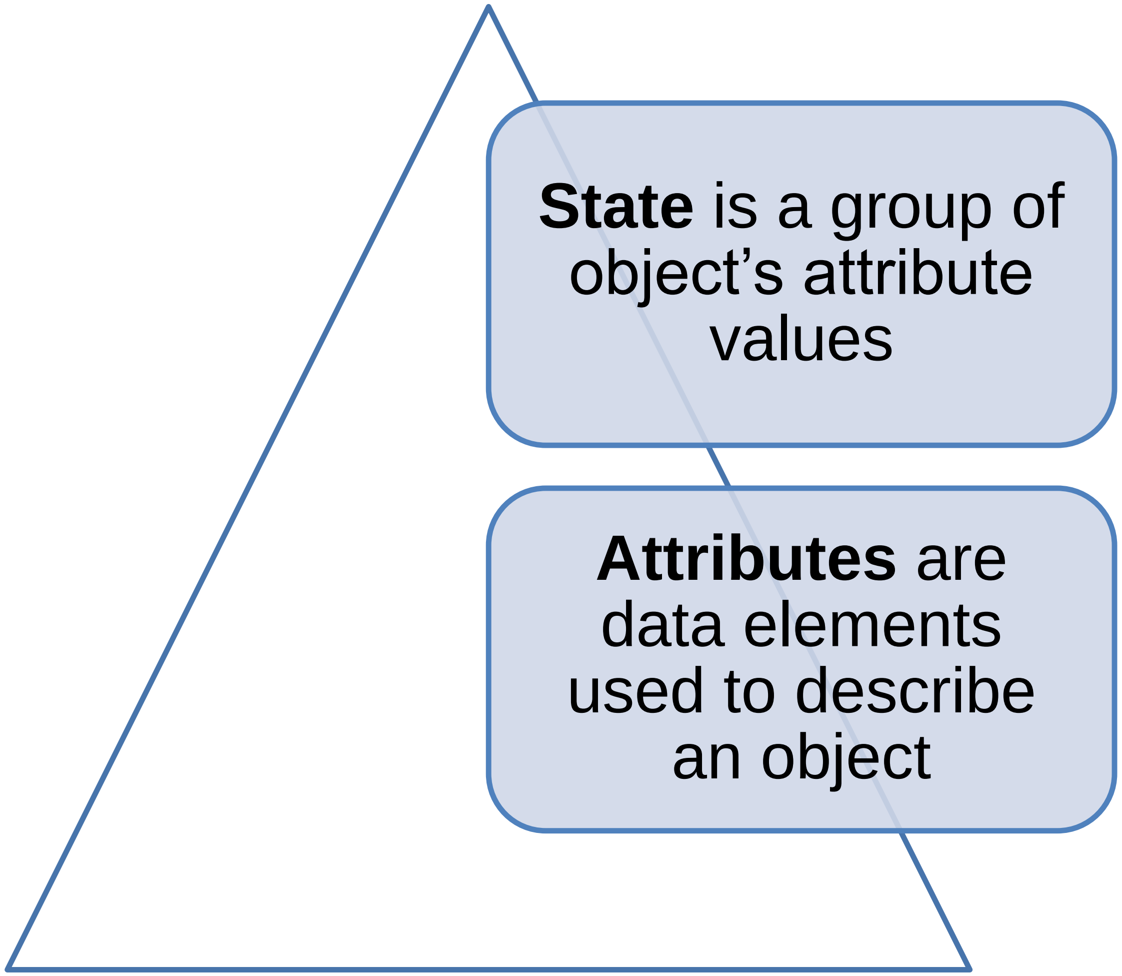
Conceptual objects

- Class
- Grade
- Age

What is a software object?



State / Data / Attributes


State is a group of object's attribute values
Attributes are data elements used to describe an object

State / Data / Attributes

Student attributes	Student state
☐ Name	☐ **Name**: Smith Garden
☐ Birth date	☐ **Birth date**: 22/JUL/1970
☐ ID	☐ **ID**: 649851
☐ Program	☐ **Program**: Computer Science

Behavior / Operations / Methods

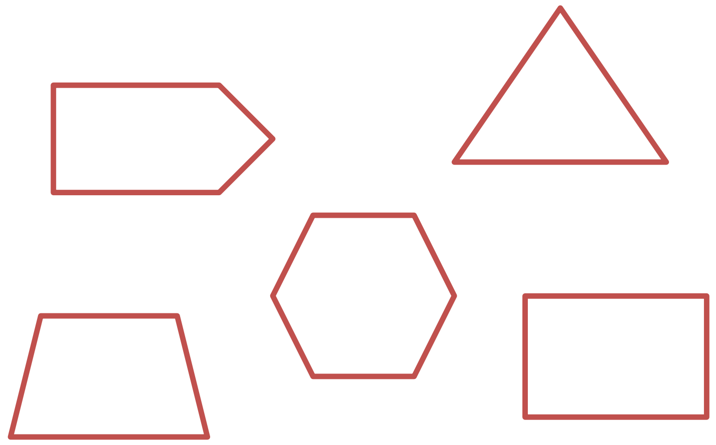
Operations are	
the things that an object does to modify its attribute values	things that an object does to access its attribute

What is a class?

Is a list of **common attributes and behaviors** for a set of similar objects.

```
Class Shape
        Area
        Perimeter
        Angles
        Edges

        Get Area
        Get Perimeter
        Get List of Angles
        Get List of Edges
```

Class example



Objects examples

3. Objects in Java

3.1 Creating classes
3.2 Instantiation
3.3 Encapsulation

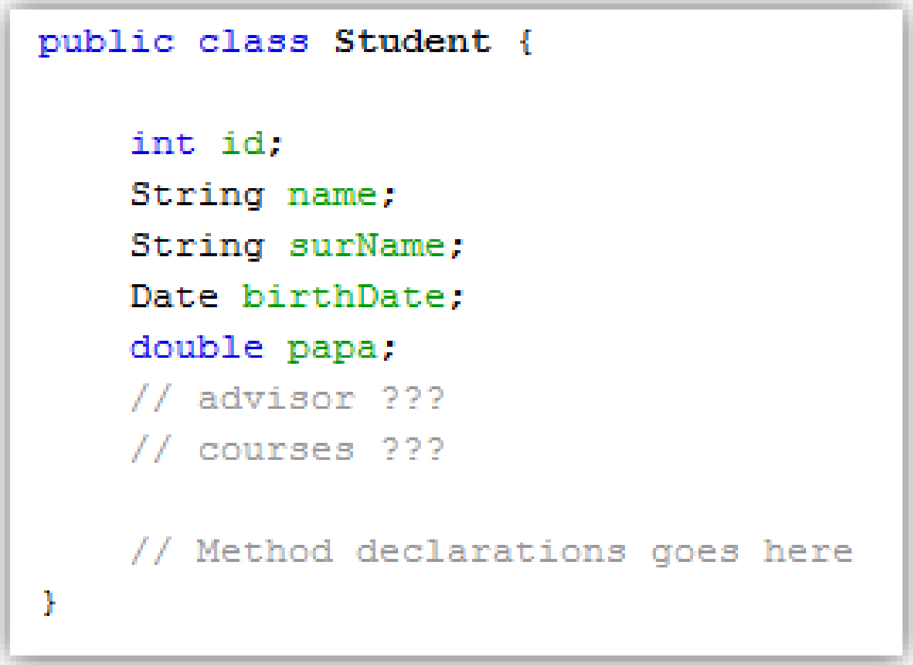
3.1 Creating classes

Creating classes

Creating a class is equivalent to defining a new object-type!

Creating the Student Class

Attribute	Type
id	integer
name	String
surName	String
birthDate	Date
papa	double
advisor	???
courses	???



```
public class Student {

    int id;
    String name;
    String surName;
    Date birthDate;
    double papa;
    // advisor ???
    // courses ???

    // Method declarations goes here
}
```

A class definition is like a class construction **template**

Instantiation

Is the process by which an object is **created in memory** based upon a class definition.

Attribute	Type	Value
id	integer	To be determined
name	String	To be determined
surName	String	To be determined
birthDate	Date	To be determined
papa	double	To be determined
advisor	???	To be determined
courses	???	To be determined

Class definition

3.2 Instantiation

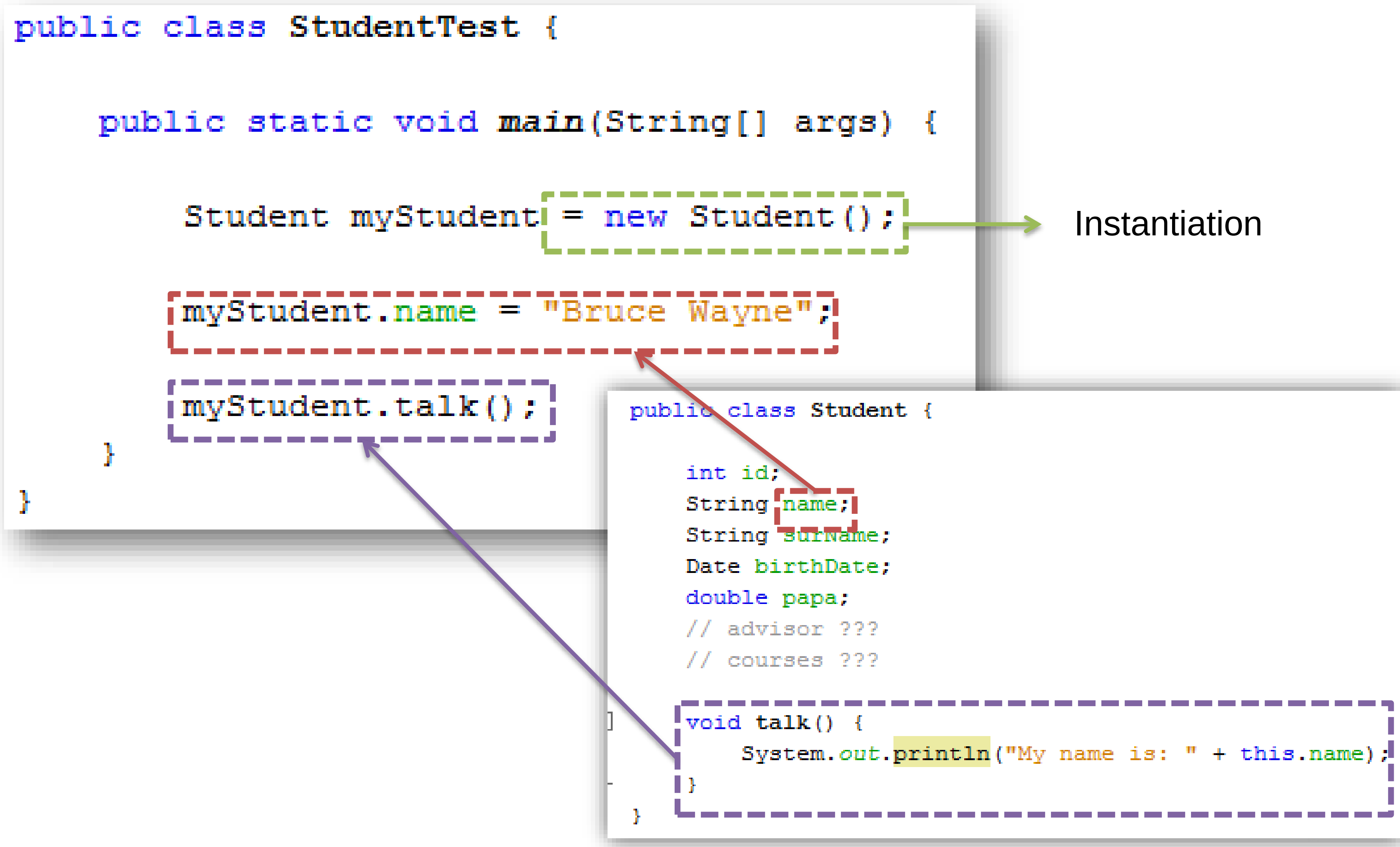
Instantiation

The process of instantiation consists of creating an object of a certain type (class).

The created object is called an instance of the corresponding class.

It refers to assigning memory to the object (with new), not the declaration!

Instantiation



```
public class StudentTest {

    public static void main(String[] args) {

        Student myStudent = new Student();

        myStudent.name = "Bruce Wayne";

        myStudent.talk();
    }
}
```

```
public class Student {

    int id;
    String name;
    String surName;
    Date birthDate;
    double papa;
    // advisor ???
    // courses ???

    void talk() {
        System.out.println("My name is: " + this.name);
    }
}
```

3.3 Encapsulation

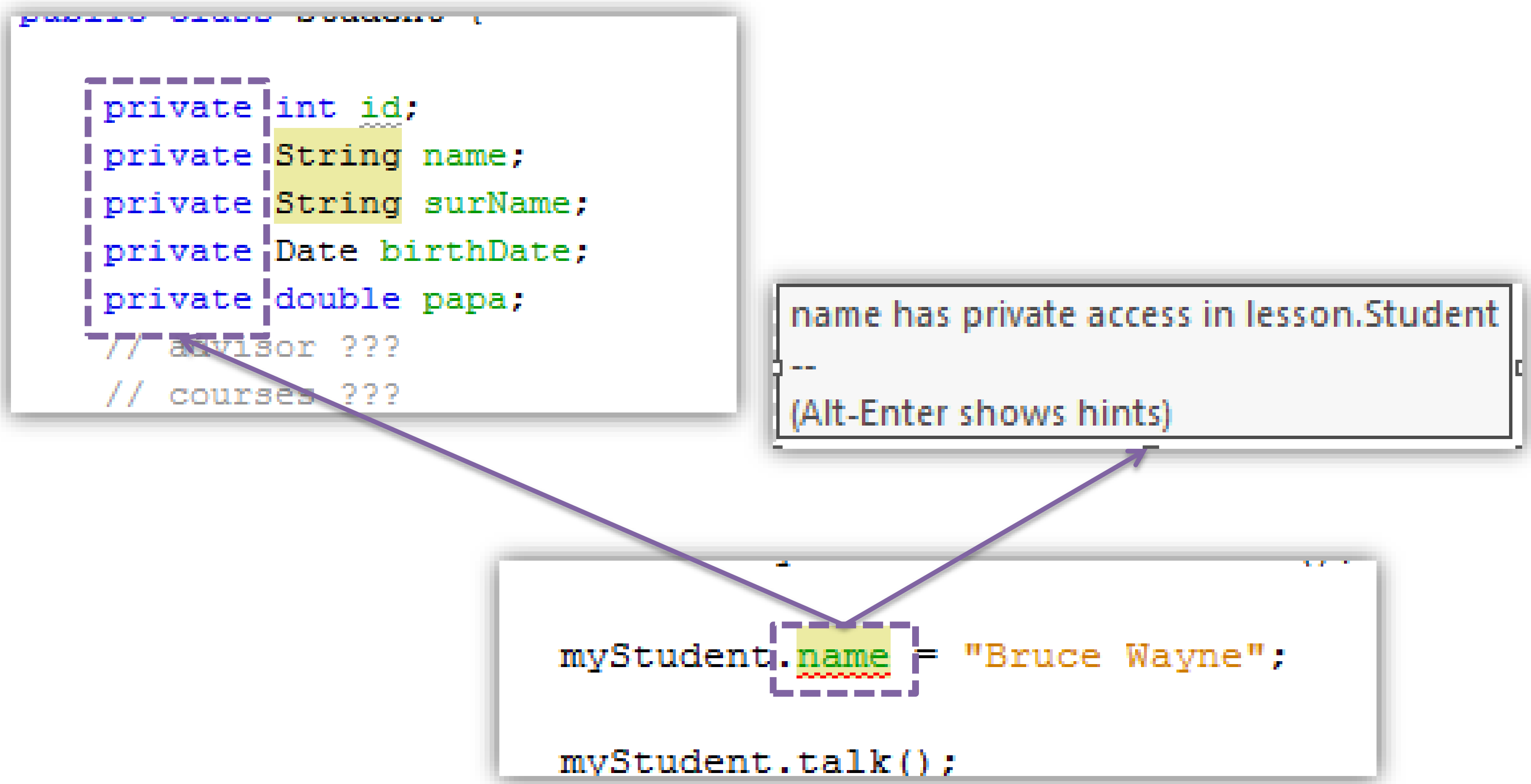
Encapsulation

Is one of the four **fundamental principles** of object-oriented programming.

Is a process of **hiding all the internal details of an object** from the outside world

Is a **protective barrier** that prevents the code and data being randomly accessed by other code or by outside the class

Encapsulation

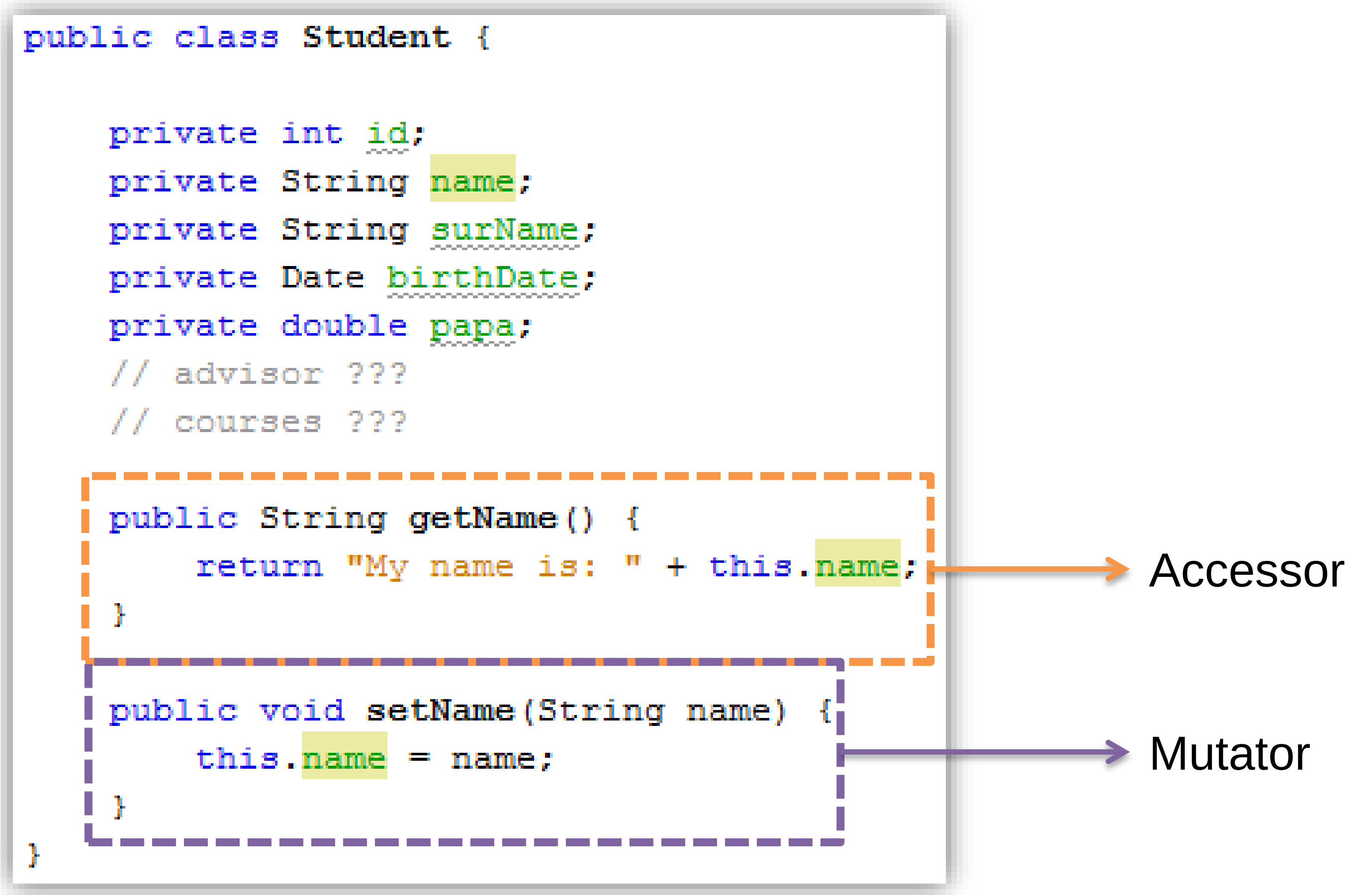


```
    private int id;
    private String name;
    private String surName;
    private Date birthDate;
    private double papa;
    // advisor ???
    // courses ???
```

```
name has private access in lesson.Student
--
(Alt-Enter shows hints)
```

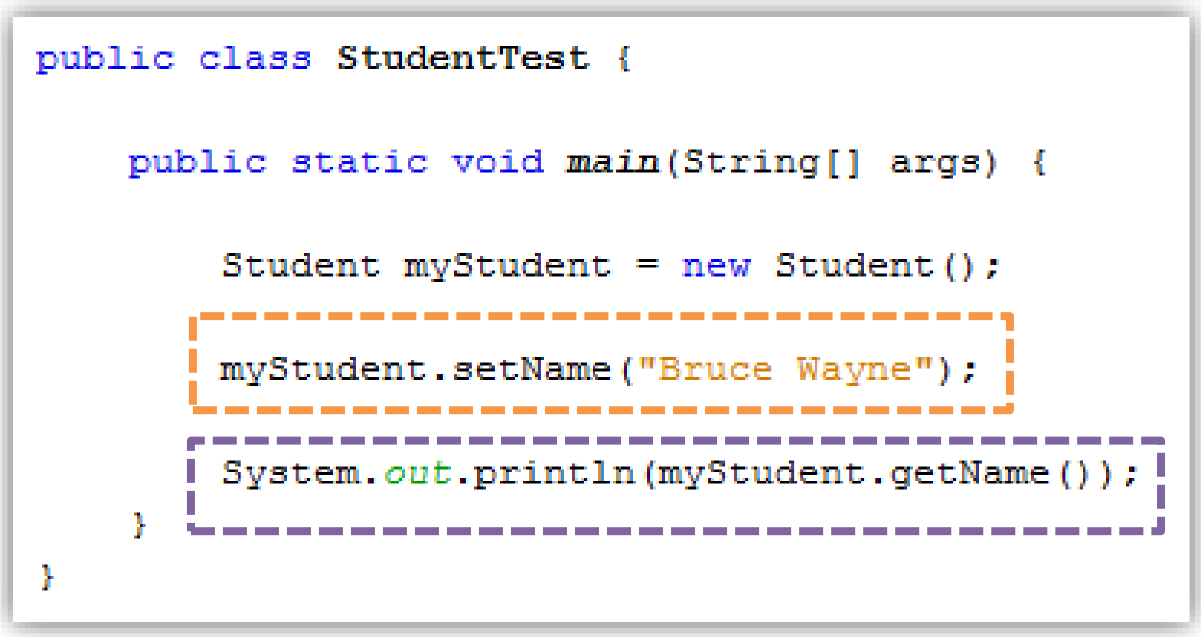
```
myStudent.name = "Bruce Wayne";

myStudent.talk();
```

Encapsulation - Accessor and mutators



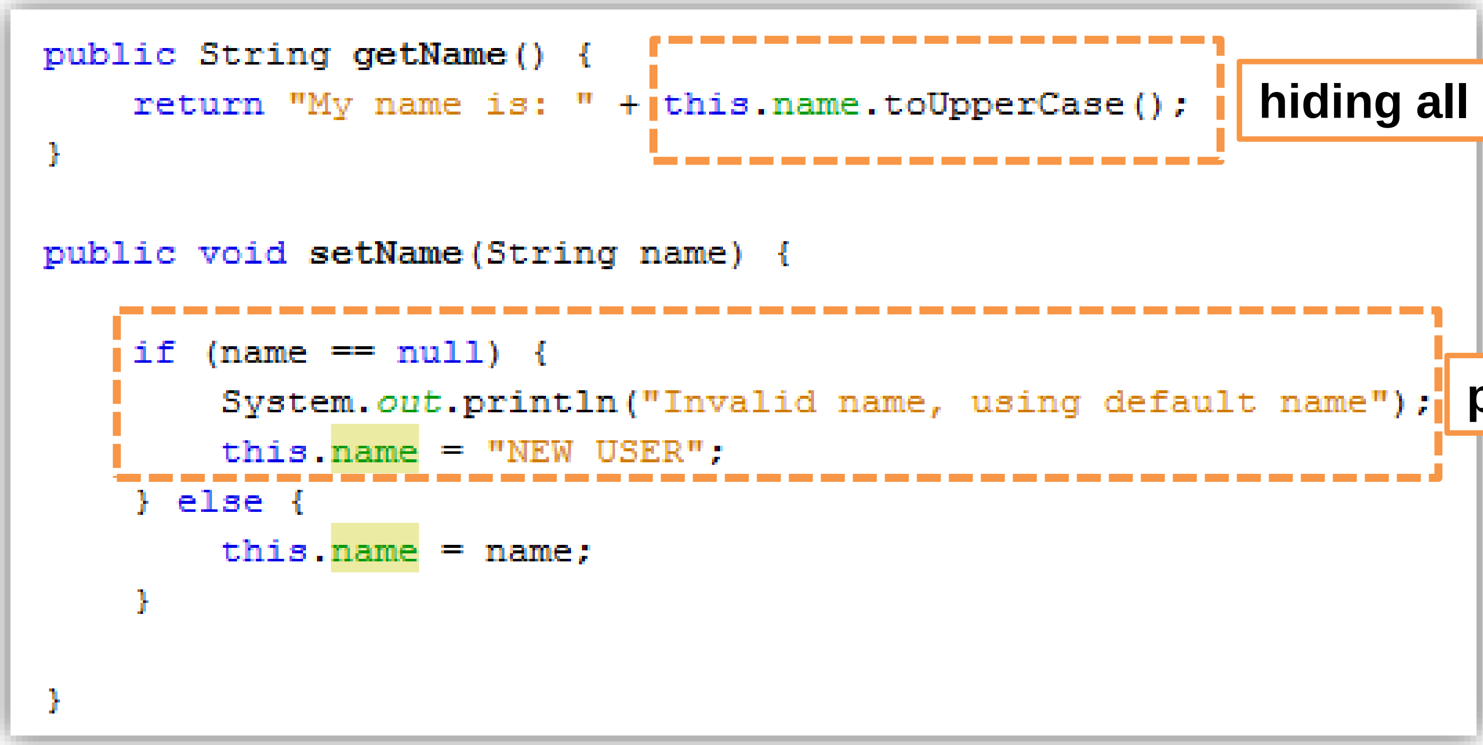
```java
public class Student {

    private int id;
    private String name;
    private String surName;
    private Date birthDate;
    private double papa;
    // advisor ???
    // courses ???

    public String getName() {
        return "My name is: " + this.name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

Using Accessor and mutators



```
public class StudentTest {

    public static void main(String[] args) {

        Student myStudent = new Student();

        myStudent.setName("Bruce Wayne");

        System.out.println(myStudent.getName());
    }
}
```

Encapsulation benefits



```
public String getName() {
    return "My name is: " + this.name.toUpperCase();
}

public void setName(String name) {

    if (name == null) {
        System.out.println("Invalid name, using default name");
        this.name = "NEW USER";
    } else {
        this.name = name;
    }

}
```

hiding all the internal detail

protective barrier

4. Instantiating Objects: A Closer Look

4.1 Working with reference variables

4.2 Garbage collector

4.1 Working with reference variables

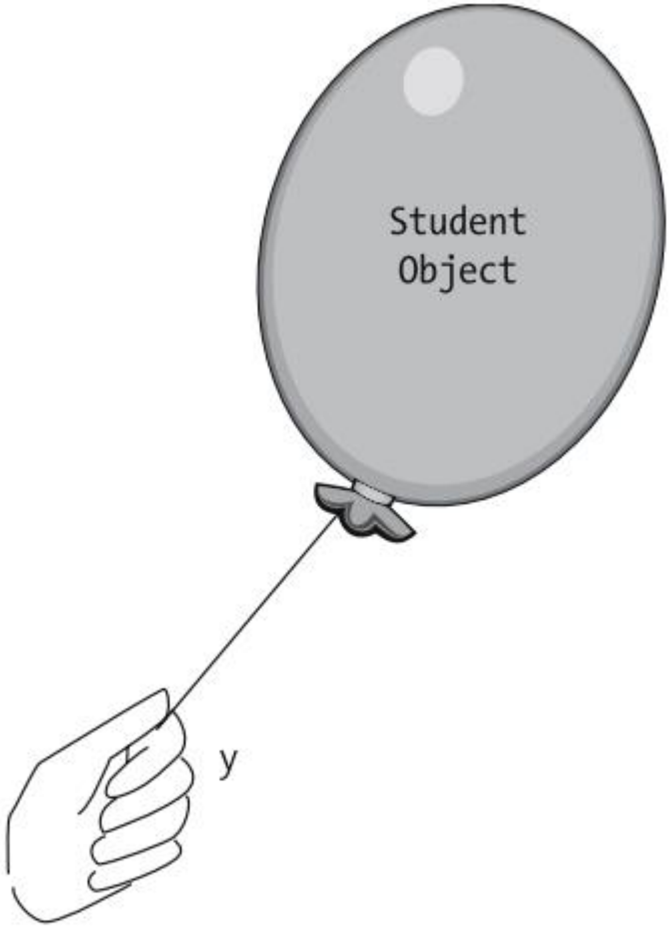
Instantiation - Working with reference variables

Object variables are also called reference variables as they do not contain the object itself but a reference to it (a reference to the position in memory where it is stored).

By default, deference variables are initialized in the value *null*, which means it is not pointing to an object yet.

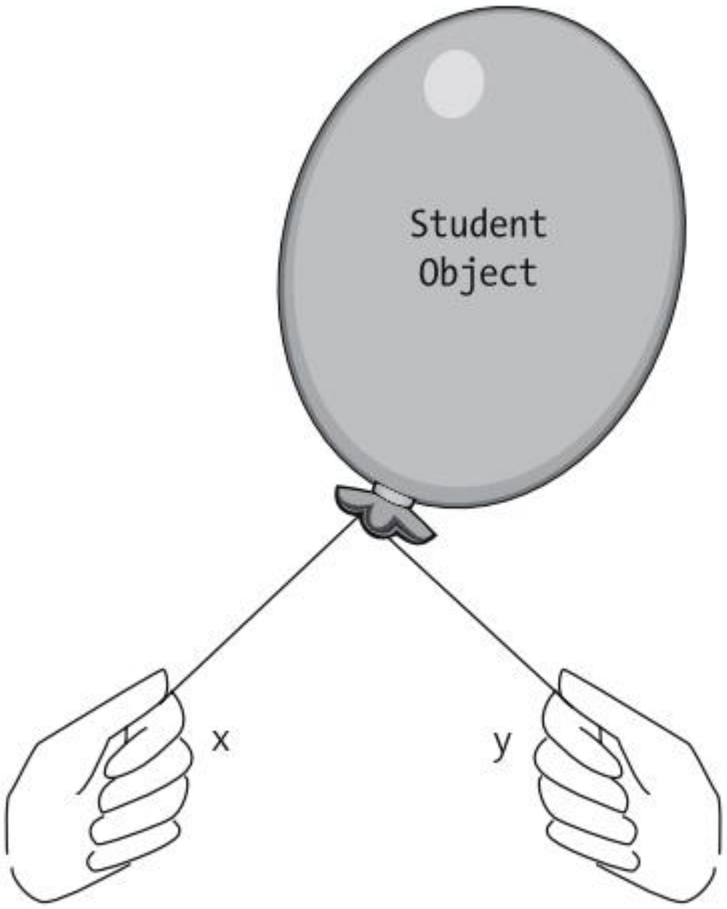
Instantiation - Working with reference variables

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

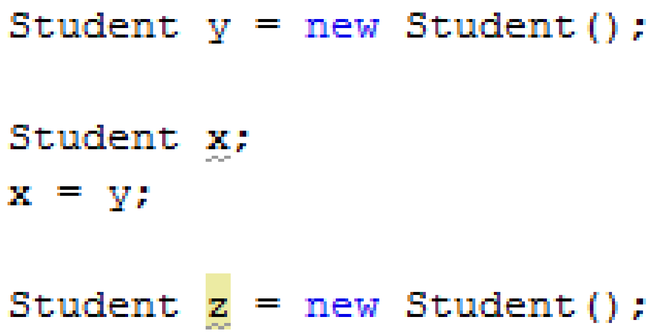


Instantiation - Working with reference variables

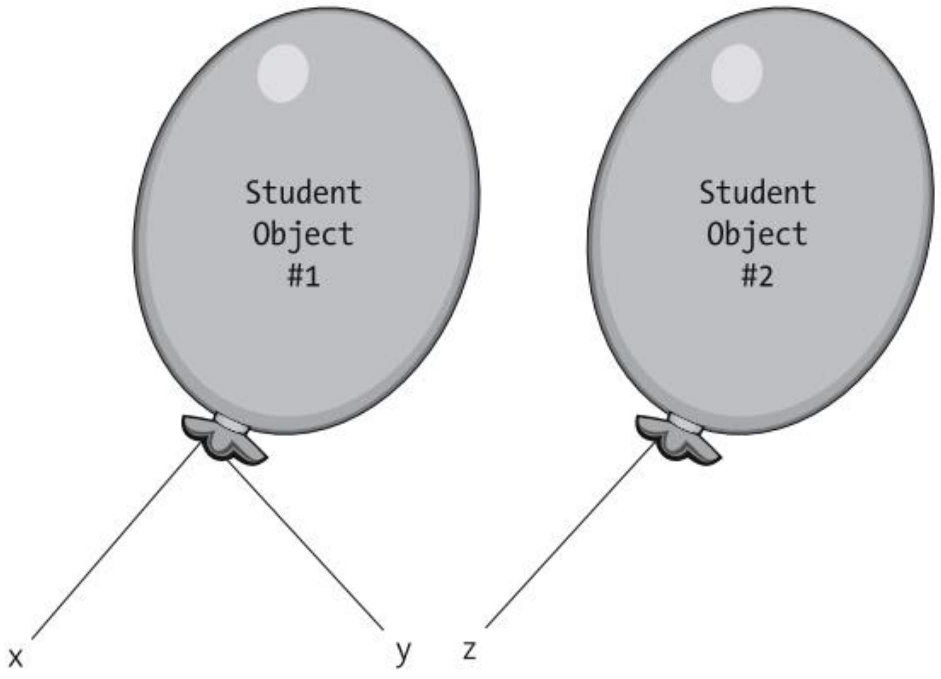
```
Student y = new Student();

Student x;
x = y;
```



Instantiation - Working with reference variables

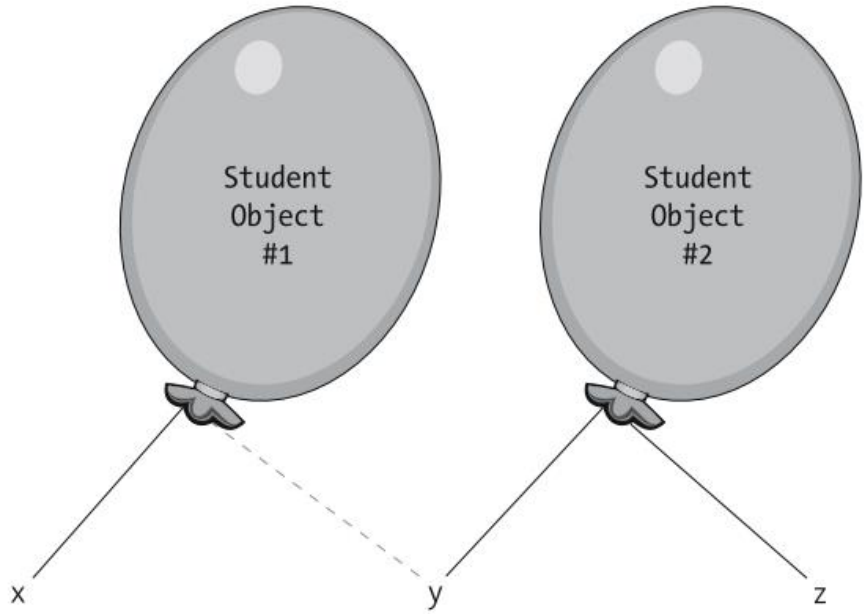


```
Student y = new Student();

Student x;
x = y;

Student z = new Student();
```



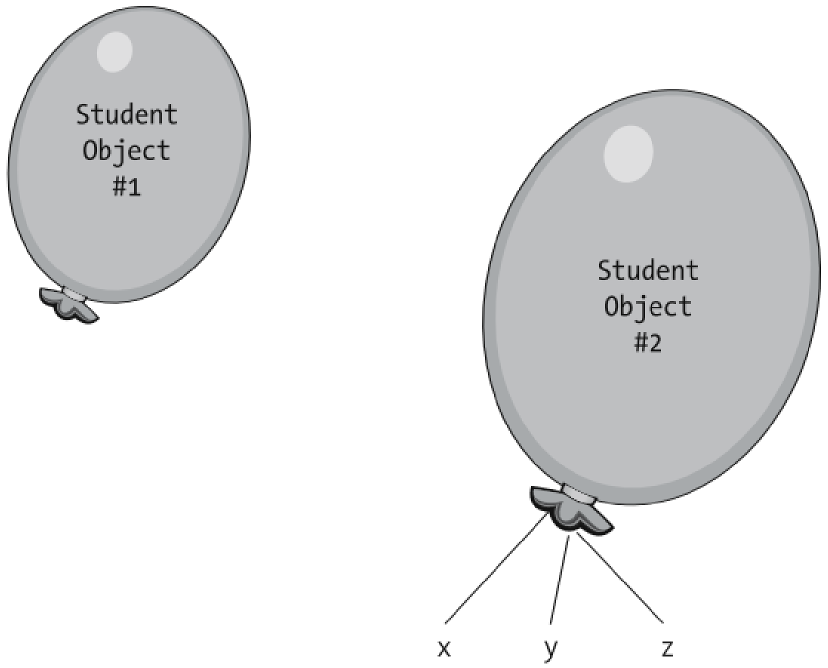
Instantiation - Working with reference variables

```
Student y = new Student();

Student x;
x = y;

Student z = new Student();

y = z;
```



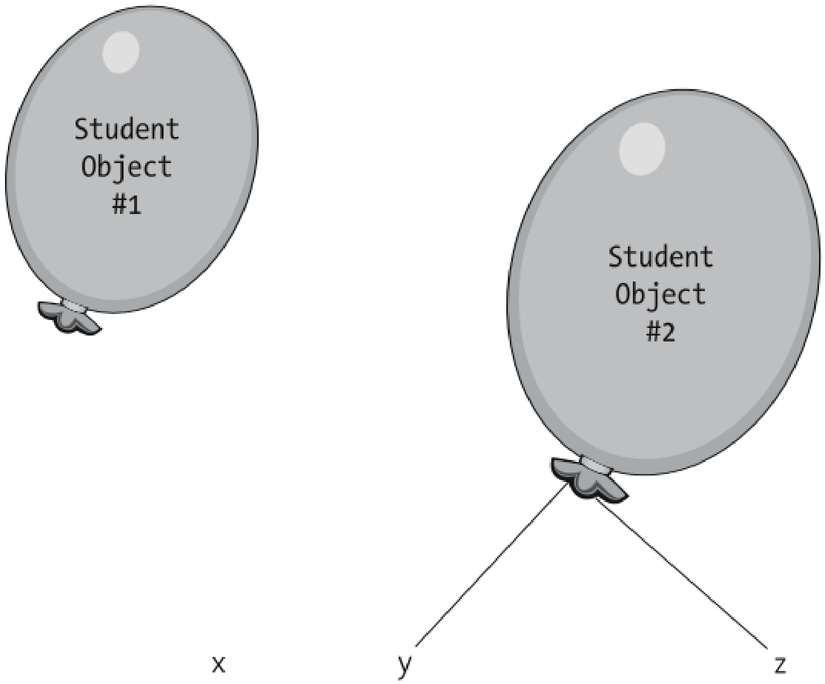
Instantiation - Working with reference variables

```
Student y = new Student();

Student x;
x = y;

Student z = new Student();

y = z;

x = z;
```



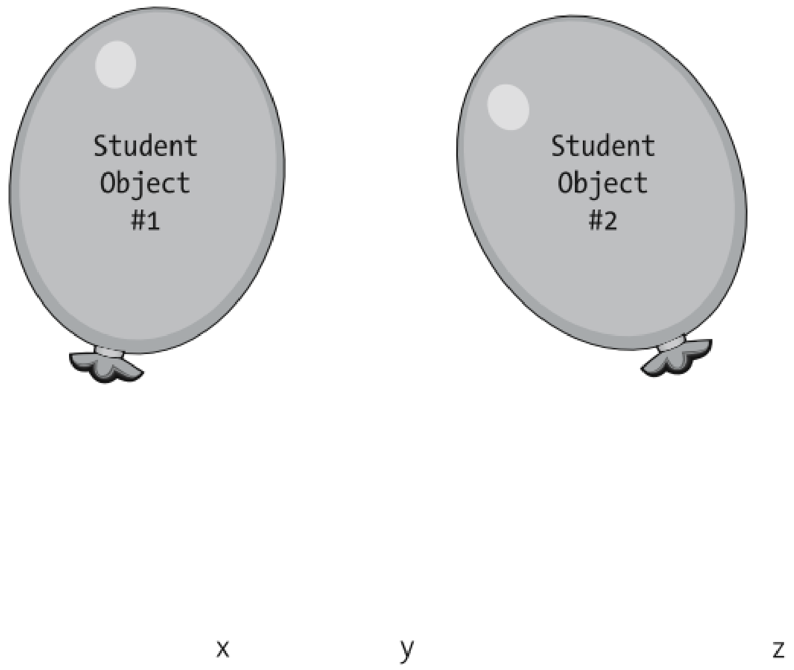
Instantiation - Working with reference variables

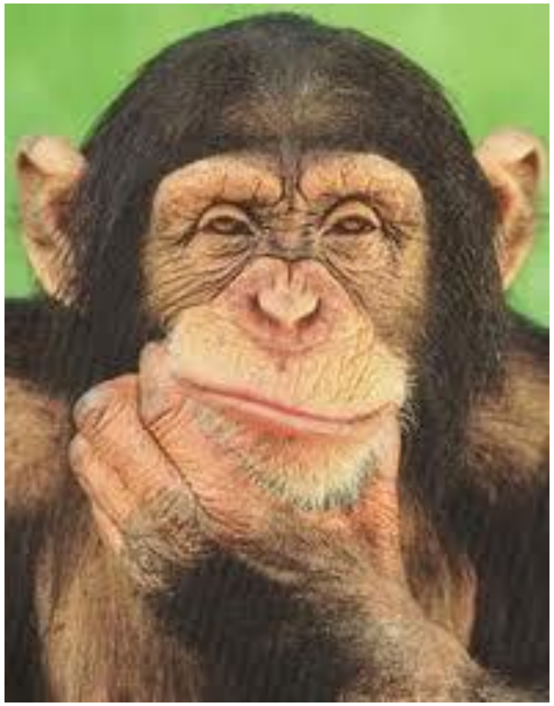
```
Student y = new Student();

Student x;
x = y;

Student z = new Student();

y = z;

x = z;

x = null;
```



Instantiation - Working with reference variables

```
Student y = new Student();

Student x;
x = y;

Student z = new Student();

y = z;

x = z;

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





Where did my balloons go?

4.2 Garbage Collector

Garbage collection

Garbage collection

- If there are no remaining **active references to an object**, it becomes a candidate for garbage collection.
- Garbage collection occurs whenever the JVM determines that the application is **getting low on free memory, or when the JVM is otherwise idle**.

5. Exercise

Class activity

1. **Abstract the model to submit the grades of a student in the Information System (Classes, behaviors, attributes, etc)**
2. Create a Java project in NetBeans or Eclipse
3. Create the Java classes of the proposed model
4. Encapsulate the classes

References

- [Barker] J. Barker, *Beginning Java Objects: From Concepts To Code*, Second Edition, Apress, 2005.
- [Deitel] H.M. Deitel and P.J. Deitel, *Java How to Program*, Prentice Hall, 2007 - 7th ed.
- [Sierra] K. Sierra and B. Bates, *Head First Java*, 2nd Edition, O'Reilly Media, 2005.