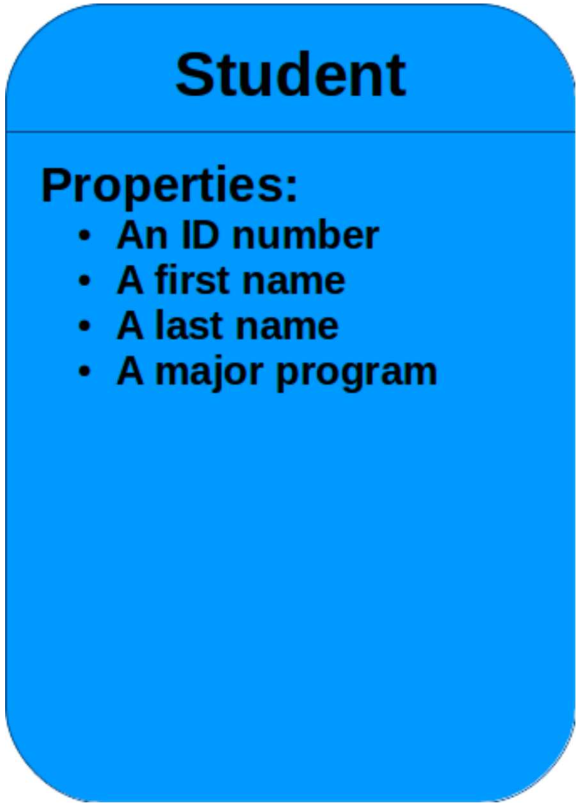
Objects: Example Applications, toString, "this" and public vs. private

What we know so far about objects and classes

Classes allow us to create complex data types, using primitive types

- Encapsulating related data in a single "type"
- i.e. pieces of information that belong together

A representation of a Student



What we know so far about objects and classes

Objects are reference types

```java
    // this is calling a constructor method
    Student s1 = new Student();
    // this is calling another constructor method
    Student s1 = Student(260412905, "Lucien", "Vil", "Library Science");
```

We are reserving `new` memory space for `s1`

`s1` is an instance of the `Student` class

```java
    // here s2 is a "null" reference
    Student s2 = null;
    // here s2 and s1 are references to the same object!
    s2 = s1;
```

What we know so far about objects and classes

To access or modify a property of an object, put a `.` after the variable name

```java
Student s1 = new Student();

s1.id = 260412905;

s1.first_name = "A";
s1.last_name = "B";

s1.major_program = "Math";

```

We call these properties attributes of a class

What we know so far about objects and classes

We can use the `Student` class as any other type

Declaring an array of elements of type `Student`

```java
    Student[] comp202_students = new Student[200];

```

Each element in `comp202_students` points an instance of the `Student` class, in the computer's memory

- Each position in the array is `null` by default
- We need to initialize each position in the array before using it

```java
Student[] comp202_students = new Student[200];

// initialize each position in the array so that it points
// to the data of a new Student
for (int i=0; i < comp202_students.length; i++){
    comp202_students[i] = new Student();
}

```

What we know so far about objects and classes

We can define methods inside a class. Inside class methods, we have acces to class attributes

```java
public class Student{
    //define class PROPERTIES here
    public int id;
    public String first_name;
    public String last_name;
    public String major_program;

    //define class METHODS here
    // A constructor method, notice it does not declare a return type
    public Student(){
        // initialize properties and execute other code by default
    }

}
```

What we know so far about objects and classes

We can define methods inside a class. Inside class methods, we have acces to class attributes

```java
public class Student{
    //define class PROPERTIES here
    public int id;
    public String first_name;
    public String last_name;
    public String major_program;

    //define class METHODS here
    // A constructor method, notice it does not declare a return type
    public Student(){
        // initialize properties and execute other code by default
    }

    // Another constructor method, notice it does not declare a return type
    public Student(int new_id, String new_first_name, String new_last_name, String new_program){
        // initialize properties and execute other code by default
    }
}
```

What we know so far about objects and classes

We can define methods inside a class. Inside class methods, we have acces to class attributes

```java
public class Student{
    //define class PROPERTIES here
    public int id;
    public String first_name;
    public String last_name;
    public String major_program;

    //define class METHODS here
    // A constructor method, notice it does not declare a return type
    public Student(){
        // initialize properties and execute other code by default
    }

    // Another constructor method, notice it does not declare a return type
    public Student(int new_id, String new_first_name, String new_last_name, String new_program){
        // initialize properties and execute other code by default
    }

    public void printProperties(){
         // Each instance can acces its own properties from a class method
         System.out.println("My student id is: "+id);
         System.out.println("My student first_name is: "+first_name);
    }
}
```

What we know so far about objects and classes

We can define methods inside a class. Inside class methods, we have acces to class attributes

```java
public class Student{
    //define class PROPERTIES here
    public int id;
    public String first_name;
    public String last_name;
    public String major_program;

    //define class METHODS here
    // A constructor method, notice it does not declare a return type
    public Student(){
        // initialize properties and execute other code by default
    }

    // Another constructor method, notice it does not declare a return type
    public Student(int new_id, String new_first_name, String new_last_name, String new_program){
        // initialize properties and execute other code by default
    }

    public void printProperties(){
         // Each instance can acces its own properties from a class method
         System.out.println("My student id is: "+id);
         System.out.println("My student first_name is: "+first_name);
    }

    public int compareTo(Student s2){
       // put some code to comapre this student with s2
       // s2 is a reference to another instance
    }
}
```

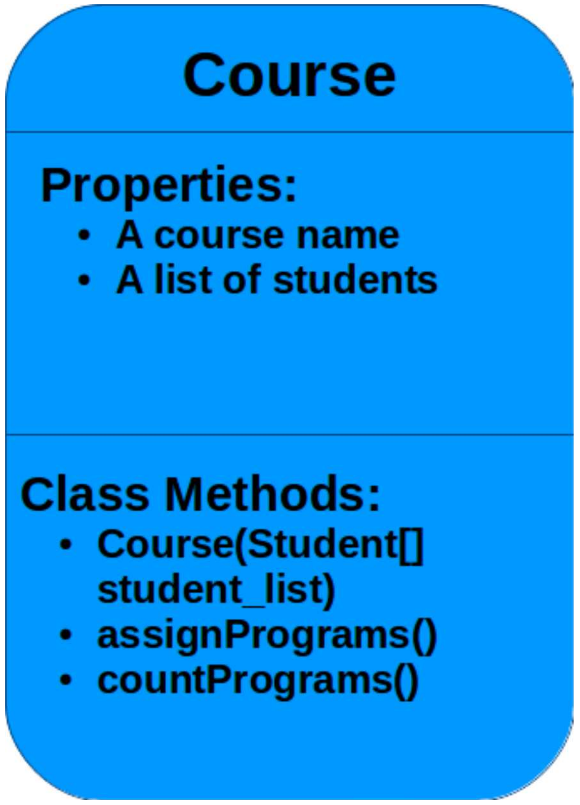
The `toString` method

All objects in java share the `toString` method. By default it returns the Memory Address of the object.

```java
public class Student{
    //define class PROPERTIES here
    ...

    //define class METHODS here
    ...

    public String toString(){
       // This method will be called when we try to PRINT this object
    }
}
```

An example - Creating a Course class

- The `Course` class
  - A `Course` has an course name, a course id, and an instructor name
  - A `Course` has a list of registered students
  - Write a method that assigns to each student a random `major_program` from { "B.A.", "B.Eng.", "B.Sc.", "B.Comm.", "M.Sc", "Ph.D" }
  - Write a method that counts how many students are enrolled in each program



The `this` keyword

The `this` returns a memory reference to the current class

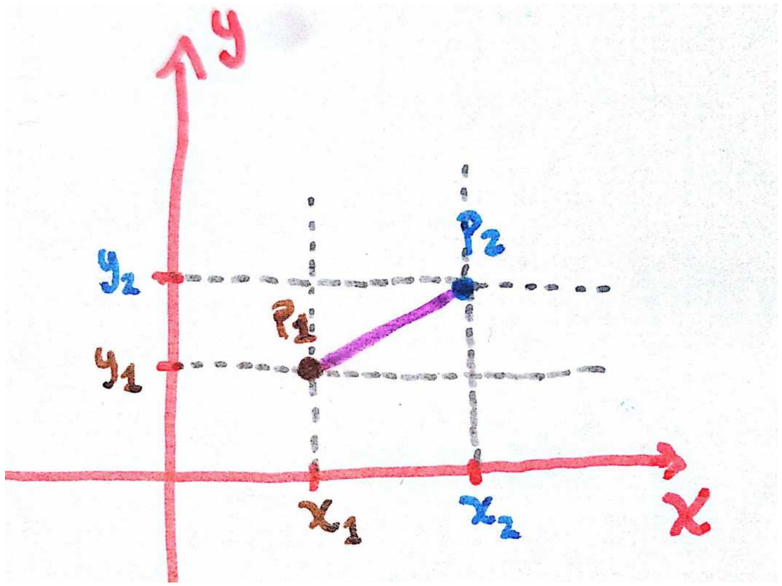
```
        public class Course{
          String course_name;
          String course_id;
          String instructor_name;
          Student[] registered_students;

          // A constructor method, notice it does not declare a return type
          public Course(String course_name, String course_id, String instructor_name){
            // initialize properties and execute other code by default

            this.course_name = course_name;
            this.course_id = course_id;
            this.instructor_name = instructor_name;
          }
      }
```

A simple exercise - Creating a 2D Point Class

- The `Point` class
  - A `Point` has two coordinates: call them `x` and `y`
  - The coordinates should be **real** numbers (`double` type)
  - Write a class method that computes the distance to another `Point`

A simple exercise - Creating a 2D Point Class

- The `Point` class
  - A `Point` has two coordinates: call them `x` and `y`
  - The coordinates should be **real** numbers (`double` type)
  - Write a class method that computes the distance to another `Point`



- sqrt((x_1 - x_2)^2 + (y_1 - y_2)^2)

Public vs Private

Methods and variables declared as `public` are accessible from any other java file

```java
public class Point{
    //define class PROPERTIES here
    public double x;
    public double y;

    // A constructor method, notice it does not declare a return type
    public Point(int x, int y){
       // initialize properties and execute other code by default
       this.x = x;
       this.y = y;
    }

    public distanceTo(Point p2){
      double dx = this.x - p2.x;
      double dy = this.y - p2.y;

      return Math.sqrt( dx*dx + dy*dy );
    }
}
```

Public vs Private

Methods and variables declared as `private` are accessible from within the class instance only

```
public class Point{
    // this variable can only be accessed by the instace reference by "this"
    private double x;
    private double y;

    // A constructor method, notice it does not declare a return type
    public Point(int x, int y){
       // initialize properties and execute other code by default
       this.x = x;
       this.y = y;
    }

    public distanceTo(Point p2){
      double dx = this.x - p2.x;
      double dy = this.y - p2.y;

      return Math.sqrt( dx*dx + dy*dy );
   }

    // getter methods
    public getX(){
        return x;
    }

    public getY(){
        return y;
    }

    public setX(double x){
        this.x = x;
    }

    public getY(double y){
        this.y = y;
    }
}
```

`private` methods and variables (members) can only be accessed through getter/setter methods

Public vs Private

Why do we care about declaring things as `private`

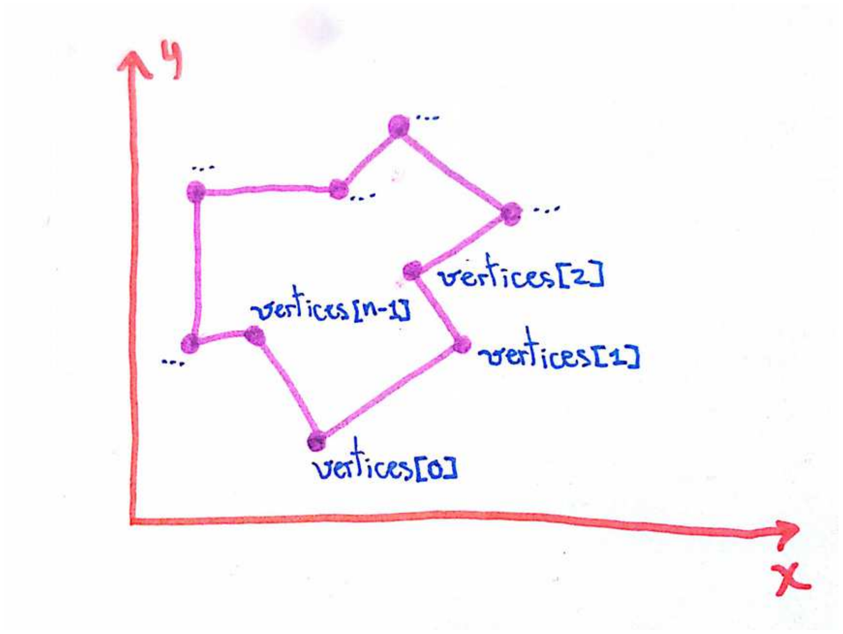
`private` methods and variables allow the designer of the code to control how the class is used (to check for correct input, to allow for future modifications of the class, to give different values depending on the state of the program, etc)

Anoter exercise - Creating a 2D Point Class, and using in a Polygon class

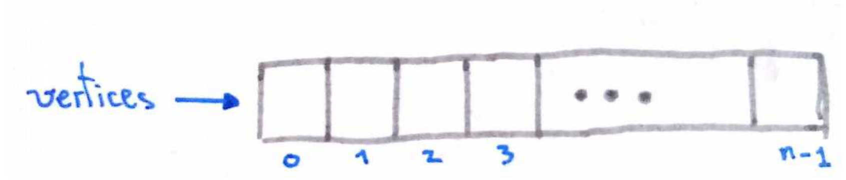
- The `Polygon` class
  - A `Polygon` has a list of `Point` objects; its vertices.
  - Write a class method that adds a new vertex to a polygon
  - Write a class method that returns `true` if the `Polygon` is **equilateral**
  - Write a class method that returns `true` if the `Polygon` is **regular**
  - Write a class method that returns the area of a polygon using the Shoelace Algorithm

Using `Point` in `Polygon`

- The `Polygon` class

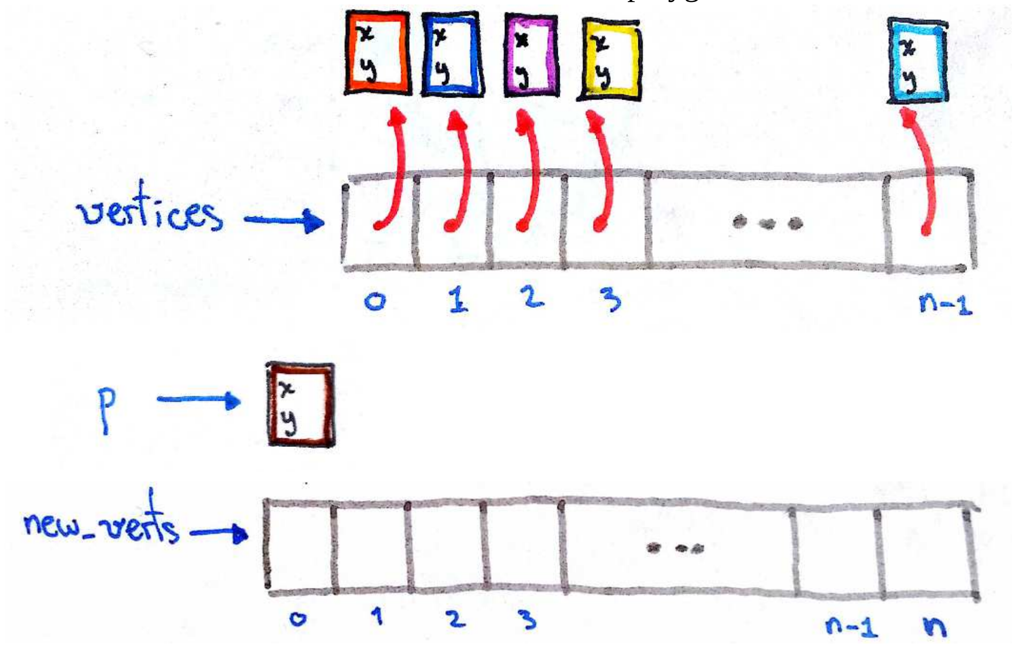


  - A `Polygon` has a list of `Point` objects; its vertices.



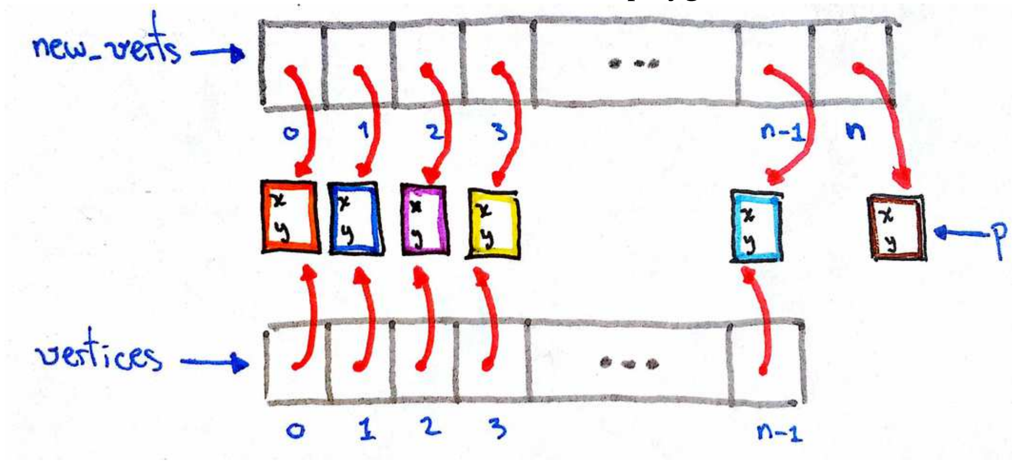
Adding a new `Point` to `vertices`

- The `Polygon` class
  - A `Polygon` has a list of `Point` objects; its vertices.
  - Write a class method that adds a new vertex to a polygon



Adding a new `Point` to `vertices`

- The `Polygon` class
    - A `Polygon` has a list of `Point` objects; its vertices.
    - Write a class method that adds a new vertex to a polygon



Completing the `Polygon` class

- The `Polygon` class
    - A `Polygon` has a list of `Point` objects; its vertices.
    - Write a class method that adds a new vertex to a polygon
    - Write a class method that returns `true` if the `Polygon` is **equilateral**
    - Write a class method that returns `true` if the `Polygon` is **regular**
    - Write a class method that returns the area of a polygon using the Shoelace Algorithm

Resources

- Classes and Objects:
  http://docs.oracle.com/javase/tutorial/java/javaOO/
- The Shoelace Algorithm:
  http://en.wikipedia.org/wiki/Shoelace_formula
- Suggested reading:
  How to think like a Computer Scientist, Chapter 11