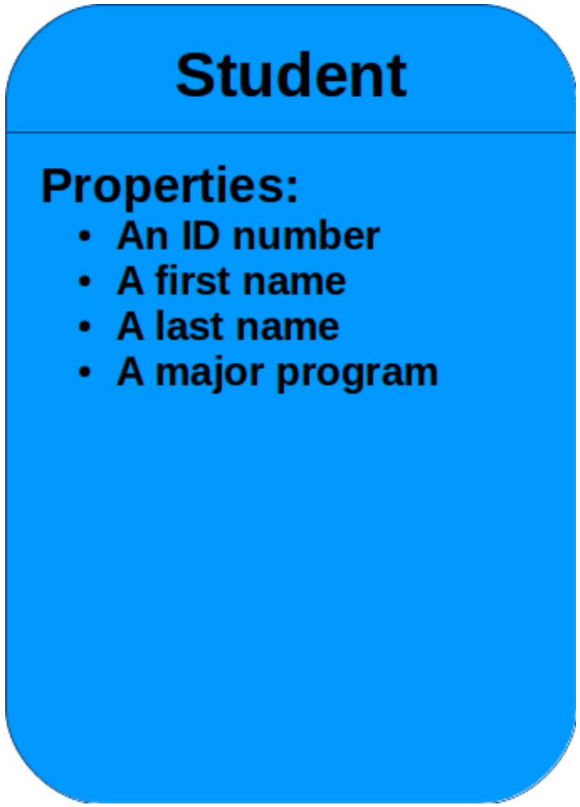
Objects: Creating your own data types

What is a class?

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



Syntax of a class

Use the `class` keyword

```
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
}

```

The `public` keyword, let's us access the properties of `Student` from outside the class definition

---

Declaring variable of type `Student`

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

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

Objects are reference types

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

Memory Address	Variable Type	Variable name	Value
@1001	Student	s1	@1100
...	...	...	...
@1100	int	id	0
@1101	String	first_name	""
@1102	String	last_name	""
@1103	String	major_program	""

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

Accessing properties of a class

Put a ． after the variable name

```
1   Student s1 = new Student();
2
3   s1.id = 260412905;
4
5   s1.first_name = "A";
6   s1.last_name = "B";
7
8   s1.major_program = "Math";
9
```

Null references

You can make a reference type point to nowhere in your memory

I.e. you can declare a reference type without the `new` statement

```
1   Student s1 = null;
2
3   int[] array = null;
4
```

Memory Address	Variable Type	Variable name	Value
@1001	Student	s1	`null`
@1001	int[]	array	`null`

No `new` memory gets reserved for `s1` or `array`

Using a class in Arrays

We can now use `Student` as any other type

Declaring an array of elements of type `Student`

```
1     Student[] comp202_students = new Student[200];
2
```

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

```
1   Student[] comp202_students = new Student[200];
2
3   // initialize each position in the array so that it points
4   // to the data of a new Student
5   for (int i=0; i < comp202_students.length; i++){
6       comp202_students[i] = new Student();
7   }
8
```

The constructor method

The constructor method

Defines some default code that is executed when we create a variable of our class

```
1   public class Student{
2       //define class PROPERTIES here
3       public int id;
4       public String first_name;
5       public String last_name;
6       public String major_program;
7
8       //define class METHODS here
9
10      public Student(){
11        // initialize properties and execute other code by default
12      }
13  }
14
```

Declaring variable of type `Student` calls the constructor method of `Student`

```
1     // this is calling the constructor method!
2     Student s1 = new Student();
```

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

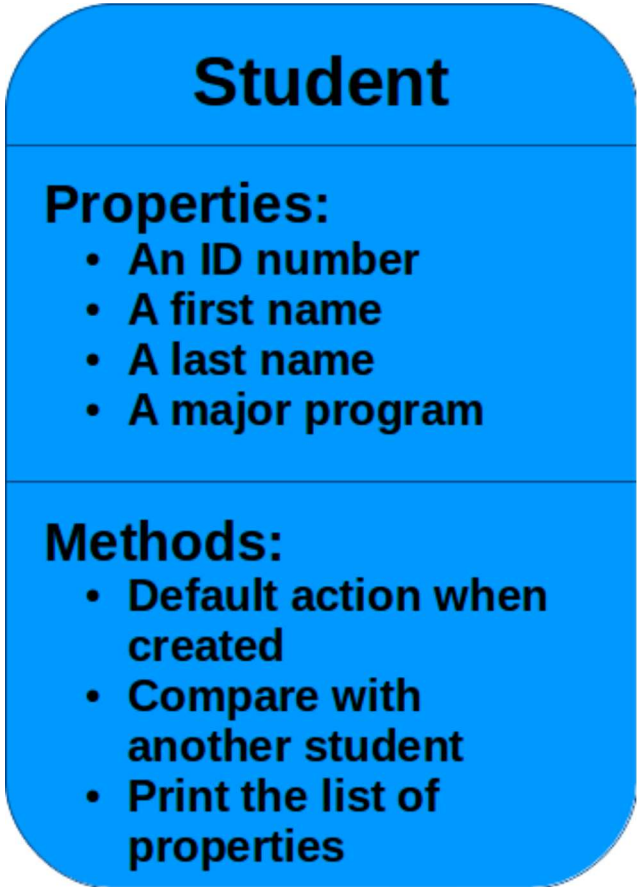
Class methods

What are class methods?

We can execute code that depends on the properties of an object

---

A representation of a Student

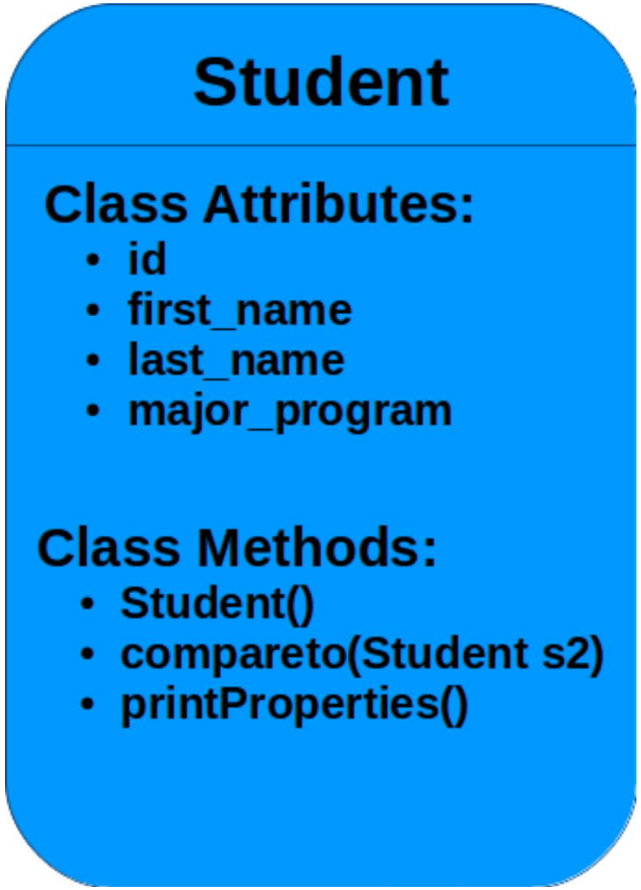


What are class methods?

We can execute code that depends on the properties of an object

---

A representation of a Student



Class method syntax

Remove the keyword `static`, from the method declaration

```
  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
      public void printProperties(){
           // some code to print stuff
      }

      public int compareTo(Student s2){
          // put some code this student with s2
      }

      // the constructor method
      public Student(){
          // initialize properties and execute other code by default
      }
  }

```

Class method syntax

Remove the keyword `static`, from the method declaration

An object instance can access its class attributes inside a class method

```
  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
      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 this student with s2
      }

      // the constructor method
      public Student(){
          // initialize properties and execute other code by default
      }
  }

```

Method overloading

Method overloading

Declaring multiple class methods with the `same name` and return type

Must have different input arguments

E.g multiple constructors

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

    // A constructor method
    public Student(){
        // initialize properties and execute other code by default
    }
}

```

Method overloading

Declaring multiple class methods with the `same name` and return type

Must have different input arguments

E.g multiple constructors

```
1   public class Student{
2       //define class PROPERTIES here
3       public int id;
4       public String first_name;
5       public String last_name;
6       public String major_program;
7
8       // A constructor method
9       public Student(){
10          // initialize properties and execute other code by default
11      }
12
13      // Another constructor method
14      public Student(int new_id){
15          // initialize properties and execute other code by default
16      }
17  }
18
```

Method overloading

Declaring multiple class methods with the `same name` and return type

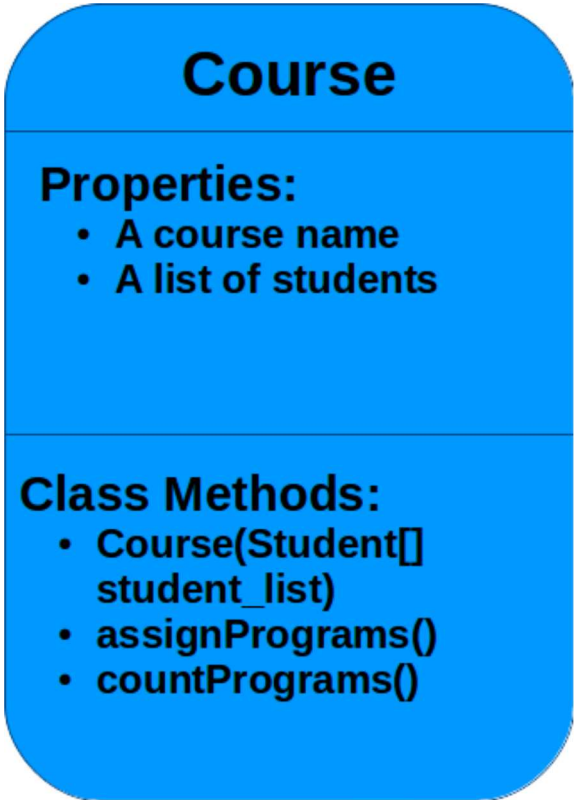
Must have different input arguments

E.g multiple constructors

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

    // A constructor method
    public Student(){
        // initialize properties and execute other code by default
    }

    // Another constructor method
    public Student(int new_id){
        // initialize properties and execute other code by default
    }

    // Yet another constructor method
    public Student(int new_id, String new_first_name, String new_last_name, String new_program){
        // initialize properties and execute other code by default
    }
}

```

An example - Creating a Course class

- The `Course` class
  - A `Course` has an course name, a course number, 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



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 harder 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

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