Objects: Creating your own data types

Announcements

1. Assignment deadline extended until Sunday October 26th at midnight
2. Midterm classroom locations posted in class website
3. The midterm is 50 multiple choice questions.
    - Read and undertand a piece of code
    - Determine the value of variables during program execution
    - Count how many times a line of code is executed inside loops
4. You can bring Letter size (A4) crib sheet, to the midterm.
    - Challenge: Draw a priate ship (and nothing else) on your crib sheet and get bonus marks.

Types of Objects we have seen in Java

Scanner: An object that processes input from the keyboard

Random: An object that generates random numbers

These object types are also known as classes

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

Example: Storing information about students enrolled in a course

Each student has

- A student ID number
- A first name
- A last name
- A Major program

Student

```
        int id;
        String first_name;
        String last_name;
        String major_program;

```

Multiple Students

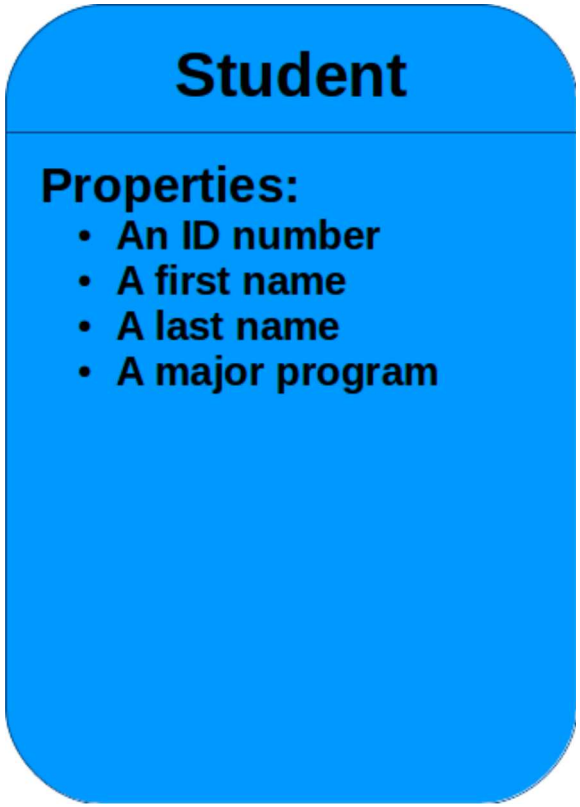
```
        int[] id;
        String[] first_name;
        String[] last_name;
        String[] major_program;

```

What is a class - 2

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 is a class - 2

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 in Java code

```
    public class Student{
        // define class PROPERTIES
    }
```

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

The constructor method

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

```
    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 Student(){
            // initialize properties and execute other code by default
        }
    }

```

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

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

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

Using a class

We can now use `Student` as any other type

Declaring a variable of type `Student`

```
1     Student s1 = new Student();
2
```

The computer will reserve some `new` space for `s1` in the computer's memory

`s1` will point to an instance of the `Student` class, in the computer's memory

Accessing properties of a class

Put a `.` after the variable name

```
    Student s1 = new Student();

    s1.id = 260412905;

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

    s1.major_program = "Math";

```

To try for yourself

Write a method that

- **receives as input two objects `s1` and `s2` of the `Student` class**
- **returns true if all of the properties of `s1` are the same as the properties of `s2`**
- **returns false otherwise**

Write a method that receives a `Student` object as an input, and prints all of its properties

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	""

Objects are reference types

```
1      Student s1 = new Student();
2
3      s1.id = 1;
4      s1.first_name = "Banana";
5      s1.last_name = "Phone";
6      s1.major_program = "Math";
7
```

Memory Address	Variable Type	Variable name	Value
@1001	Student	s1	@1100
...	...	...	...
@1100	int	id	1
@1101	String	first_name	"Banana"
@1102	String	last_name	"Phone"
@1103	String	major_program	"Math"

Objects are reference types

```
1     Student s1 = new Student();
2
3     s1.id = 1
4     s1.first_name = "Banana";
5     s1.last_name = "Phone";
6     s1.major_program = "Math";
7
8     Student s2 = new Student();
9
```

Memory Address	Variable Type	Variable name	Value
@1001	Student	s1	@1100
@1002	Student	s2	@1200
...	...	...	...
@1100	int	id	1
@1101	String	first_name	"Banana"
@1102	String	last_name	"Phone"
@1103	String	major_program	"Math"
...	...	...	...
@1200	int	id	0
@1201	String	first_name	""
@1202	String	last_name	""
@1203	String	major_program	""

References point to locations in the computer memory

References are what your computer uses to identify a variable

```
    Student s1 = new Student();

    s1.id = 1
    s1.first_name = "Banana";
    s1.last_name = "Phone";
    s1.major_program = "Math";

    Student s2 = new Student();

```

In the previous example, `s1` points to memory location `@1100`, and `s2` points to memory location `@1200`

Your computer uses `@1100` and `@1200` to identify/index/address `s1` and `s2`

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     // will this call the constructor for each position in the array?
2     Student[] comp202_students = new Student[200];
3
```

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

Resources

- Classes and Objects:
  http://docs.oracle.com/javase/tutorial/java/javaOO/
- Suggested reading:
  How to think like a Computer Scientist, Chapter 11

Drawing Tools