

CSE 11

Accelerated Intro to Programming

Lecture 1

Greg Miranda, Summer 1 2021

This lecture is being recorded

Fair Notice of Class Recording Announcement

- Each class online lecture for CSE 11, including this one, will be recorded and made available to students asynchronously.

About Me and My Family :)

Full name: **Gregory Miranda**

Preferred name: **Greg**

Preferred pronouns: **he/him/his**

Personal webpage:

<https://gregmiranda.com>



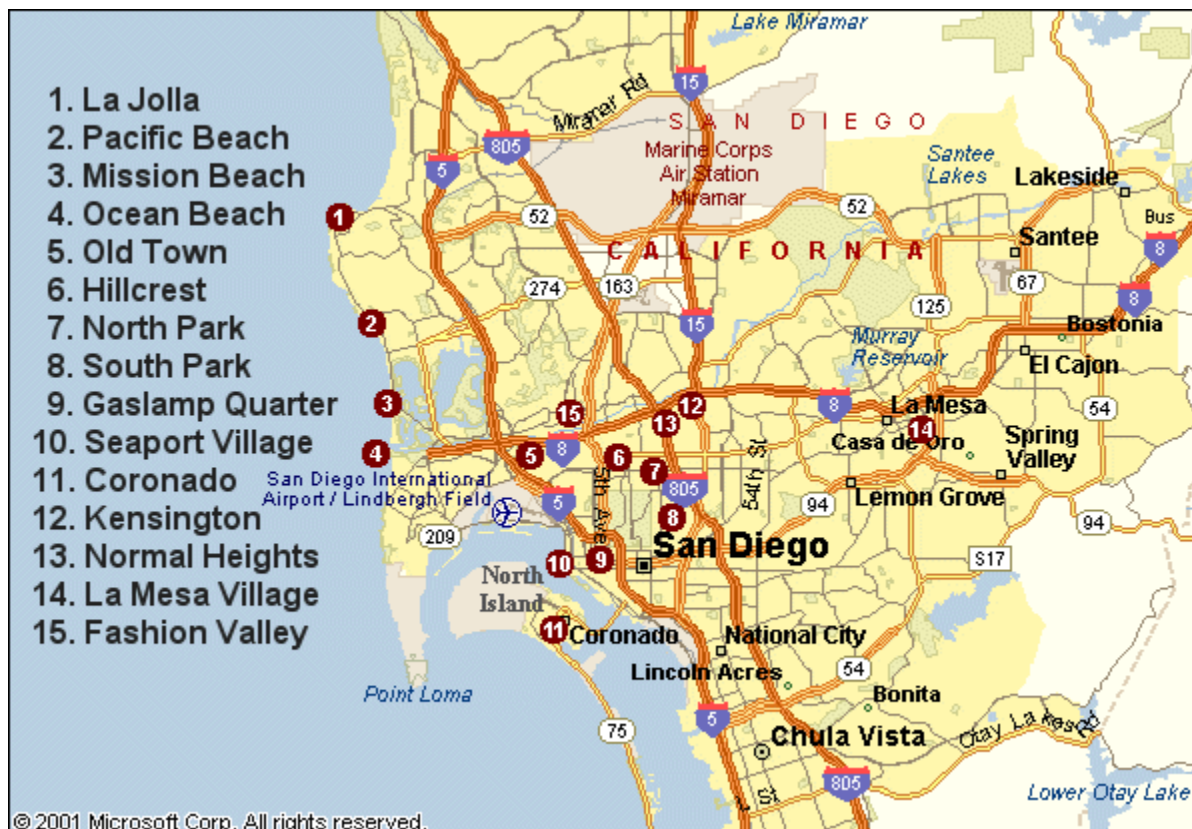
UCSDCSE
Computer Science and Engineering

Education

M.S. in Technology (Artificial Life Programming)

B.S. in Computer Engineering

Where am I from



Teaching Experience

UCSD

- CSE 3 – Fluency in Information Technology
- CSE 5A – Introduction to Computer Science (C)
- CSE 8B – Introduction to Programming and Computational Problem-Solving II
- CSE 11 – Introduction to Programming and Computational Problem-Solving:
Accelerated Pace
- CSE 12 – Basic Data Structures and Object-Oriented Design

Other universities

- Game Programming, Game Design, Web Programming

Fun things!



Announcements

- Discussion starts today @ 2pm
- PA0.5 released yesterday – due Thursday
- PA1 released today – due Thursday

Coding Experience

- How much coding experience?
 - A – No coding
 - B – A little bit of coding
 - C – Some coding
 - D – Lots of coding

Coding Experience

- Languages – Select All
 - A – Java
 - B – C++
 - C – Python
 - D – JavaScript or other scripting language
 - E – Other compiled language (ex: C)

Topics

- Syllabus
 - <https://ucsd-cse11-su121.github.io/>
- Canvas
- Questions?

Experimenting with running Java Programs

- How to edit Code?
 - Text Editor
 - Integrated Development Environment (IDE)
- How to run Code?
 - Command Line
 - Mac/Linux
 - Windows

```
javac -cp tester.jar *.java  
java -classpath tester.jar:. tester.Main Example1Lecture
```

```
javac -cp tester.jar *.java  
java -classpath tester.jar;. tester.Main Example1Lecture
```

```
class Examples1Lecture {  
    int theNumberFive = 2 + 3;  
}
```

- Fields
- Arithmetic Expression
- Output
- Java / Programming Languages
 - Simplest thing you can do with them: calculators
- More Examples

Errors / Error Messages

- Big part of programming
 - Understanding when you made a mistake
 - How to fix the mistake
- Possible mistakes
 - Invalid command
 - Expect that that class is defined in a file with the same name .java
 - Class can't be found – typo in the name or a has mismatch with the name of the class
 - Name of field doesn't match value
 - Pick meaningful names

- Possible mistakes (cont.)
 - Could leave off or forget int
 - Syntax error
 - Error messages do not always match what's wrong
 - Use context of program to figure out what happened
 - Other errors we can get
 - When running programs – practice trying to break them a little bit
 - Remove =
 - Remove ;
 - Remove { or }
- Going to be a lot of times where you make a mistake
 - Typo
 - Copy/paste incorrectly
 - Accidentally delete something
 - Or just make a mistake
- Need to practice fixing error messages
 - Use the context of the program to understand the error message
- Errors are a normal part of programming

Arithmetic in Java

```
class Examples1Lecture {  
    int x = 2 + 9 * 3;  
    int y = 10 / 3;  
}
```

- What's would happen here?
 - What is the value of x?
 - What is the value of y?
 - What about `y = 11 / 3; ?`
- Using parenthesis

- Order of operations & parenthesizing
 - In many ways Java acts like arithmetic
 - But in other ways, Java does not
 - Division has truncation behavior we do not see in math classes
 - Very common in programming
 - Multiplication & division before addition & subtraction

New Example

- Create a new file

```
class Examples2 {  
    int rate = 20;  
}
```

- class – will talk about more later
 - For now: describes a group of fields
- Problem:
 - Calculate the pay you would receive at a certain hourly rate given a number of hours
 - New field: # of hours worked
 - Calculate total pay using Java as a calculator

- Calculate total pay (cont.)
 - Can we use these fields in another calculation?
 - Why is it useful to do this with fields instead of writing this directly?
 - What if:
 - Use same hourly rate, but a number of different weeks to calculate?
 - What if:
 - We want to change the hourly rate?
 - Change once, changes all values
 - Many times, you will have one field whose value can be used in many places
 - Configure how the program works
 - Changing the value in one spot can affect many other places in the program
 - Powerful concept in programming:
 - Define a value in one place
 - Change it by editing the program
 - Watch its changes be reflected in all the other places next time it's run

- Using `this.hourlyRate`
 - Call that a field look-up or a field access
 - Looking up the current value of a field that has been defined before

Text

- Integers (int) – common kind of data programmers work with
- New kind of data – also really common – text
 - Examples: usernames, passwords, email, names, addresses
 - Data type for text - String
- Previous examples - had int as the type
 - `int numberOfStaff = 14;`
- Now – using String as the type
 - `String name = "Greg Miranda";` //String value, string literal
- `String className = 11;`
 - What happens? Does it work?
- `String className = "11";`
 - What happens? Does it work? Is it text or a number?

Types

- int – integer type – integer literal
- String – text type – string literal (written in double quotes)
- Java will enforce that we always
 - store string values in String typed fields
 - numeric values in numeric typed fields
- Programmer's job to get this right
 - Java will give an error message if we don't

String

- We learned we can store Strings values in fields
 - What else can we do with them?
 - Can we add Strings together, like integers?
 - `String fullName = "Greg" + "Miranda";`
 - Will this work?
 - Can we multiply Strings by a number?
 - `String str = this.firstname * 2;`
 - What about Divide? Subtract?
 - What about +? Can we add a String and a number?
 - `String str = this.firstname + 2;`
 - What's going to happen if we try this?
 - Compiler error?
 - Works? If it works, what does it store in the str field?

- We can + other things besides numbers to Strings and get similar behavior
 - More on this in upcoming weeks
- Adding Strings and numbers
 - Can be convenient
 - Can turn a number into text
 - Can also be confusing
 - `String className = "11" + 200;`
 - `int className = 11 + "200";`
 - Error
 - `String className = 11 + "200";`
 - Java does do this automatic conversion of Strings and numbers
 - Be careful in your own code

Vocabulary

```
class Example {  
    int x = 3 + 2;  
    int y = this.x * 4;  
}
```

How many field definitions are in this class?

```
1 class C {  
2     int a = 10;  
3     String b = 5 + "A";  
4 }
```

How many field definitions are in this class?

```
1 class D {  
2     int a = 10;  
3     String b = this.a + " dollars";  
4 }
```

Do you think there's a limit on how many field definitions can be in a class?

Program Steps

```
class Example {  
  int x = 3 + 2;  
  int y = this.x * 4;  
}
```

Expressions

- `int x = 3 + 2;`
 - `3 + 2`
 - Arithmetic expression
 - Binary operator expression
- `int y = this.x * 4;`
 - `this.x`
 - Field access expression
 - `this.x * 4`
 - Arithmetic expression where the left-hand operand is a field access expression

Methods

- New class – MethodExample
- In programming, we often want to describe a computation once
 - Then reuse it on different numbers, or different values
 - Write once, use it over and over again
- Example:
 - Take two numbers and add up their squares
 - `int sos1 = 3 * 3 + 5 * 5;`
 - `int sos2 = 4 * 4 + 7 * 7;`

- Define a method to do the same thing

```
int sumSquares(int n, int m) {  
    return n * n + m * m;  
}
```

- Vocabulary:
 - Method definition
 - Parameters
 - Method body
 - return keyword

- Running it...
 - Method definition doesn't change what prints out or any of the fields
 - Run command – only prints out the values of the fields
- Can use sumSquares() to do the calculation
 - `int ans1 = this.sumSquares(3, 5);`
 - `int ans2 = this.sumSquares(4, 7);`
- Vocabulary:
 - Called the method
 - Arguments

- Methods: one of the building blocks for building programs
 - Not just useful for arithmetic
 - Useful for many more things
- Why do we care about methods?
 - Methods give us a centralized place to write a calculation
 - Change in one place, every place that uses the method will see that update
 - As program gets large:
 - Might have 100s of places where we want to use a formula or calculation
 - Update them all by changing one place
 - Methods are self documenting – with meaningful names

```
class MethodExample {  
  
    int sumSquares(int n, int m) {  
  
        return n * n + m * m;  
  
    }  
  
    int ans1 = this.sumSquares(3, 5);  
  
    int ans2 = this.sumSquares(4, 7);  
}
```

Method definition

```
class MethodExample {  
    int sumSquares(int n, int m) {  
        return n * n + m * m;  
    }  
    int ans1 = this.sumSquares(3, 5);  
    int ans2 = this.sumSquares(4, 7);  
}
```