Introduction to Computer Science I
ICS 111

Teacher: Edoardo Biagioni (“Edo”, pronounced as in: 江戸 , Lar**edo**, Alfr**edo**)

Teaching Assistants:

- (someone that prefers that google searches for his name no longer return this page), Sections 2 and 6
- Jesse McDonald, Sections 1 and 5
- Jeremy Ong, Sections 3 and 4

ICS 111 Format
Fall 2020

- Unconventional Semester
  - **initially online only**, until further notice
- Support both **online** students and **in-person** students
  - anyone can attend entirely online
  - those who learn better in person are welcome in person
- In-person attendance: must follow all UH rules at https://manoa.hawaii.edu/covid19/guidelines/personal-safety-practices/
  - stay home if sick or exposed! wear a mask!
  - instructor/TA must record attendance
  - can only attend lectures with your section
  - at most one lecture of every six
  - at most one lab per week

ICS 111 Format: Lectures

- Content is pre-recorded and students **must** listen to it no later than **noon the day before the lecture**
- In-person lecture time is for interaction and a daily quiz
  - quiz made available online, **students must use own device**
- In-person lecture time will be zoomed, **attendance** (online or in person) **is required**
- dates when students may attend lecture in person:
  - section 1 (CRN 73054): only on aug 24, sep 16, oct 7, oct 28, nov 23
  - section 2 (CRN 73055): only on aug 26, sep 21, oct 12, nov 2, nov 25,
  - section 3 (CRN 74019): only on aug 31, sep 23, oct 14, nov 4, nov 30,
  - section 4 (CRN 73516): only on sep 2, sep 28, oct 19, nov 9, dec 2,
  - section 5 (CRN 74822): only on sep 9, sep 30, oct 21, nov 16, dec 7,
  - section 6 (CRN 74823): only on sep 14, oct 5, oct 26, nov 18, dec 9

ICS 111 Format: Labs

- Each in-person lab is also made available online via zoom or discord
- Students with a UH student ID ending in 0-4 may come to lab in person only on Tuesdays
- Students with a UH student ID ending in 5-9 may come to lab in person only on Thursdays
- Lab attendance is not required

ICS 111 Format: Assignments

- made available on Fridays
- due the following Friday at noon
- e.g. first assignment available on August 28th, due September 4th by noon
- must be done individually

ICS 111 Format: Lecture Reports

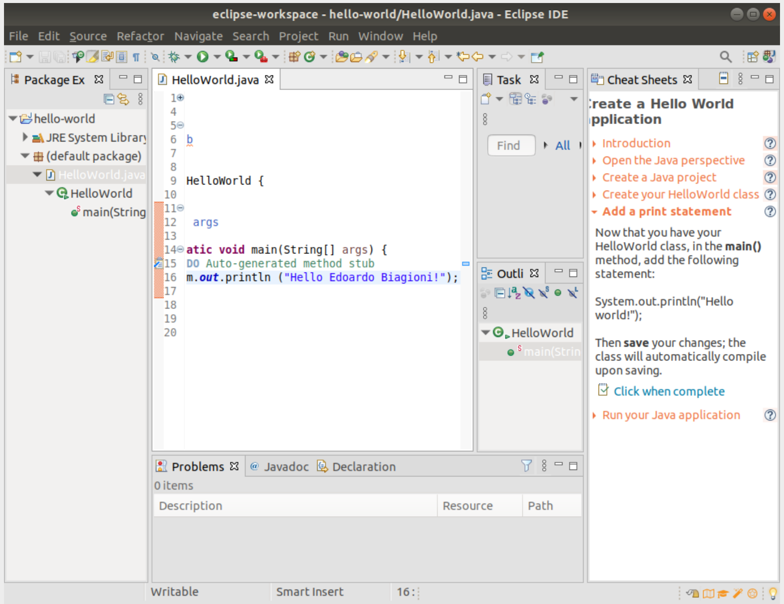
- Listen to the recorded lecture material
  - made available on Fridays and Mondays
- Write a one-paragraph (400-600 characters) report of the lecture content
- Must be done individually
- Graded on whether the student actually listened to the material
- Questions are welcome
- Due by 3pm the day before the lecture (Sun/Tue)
  - welcome before Sunday too...

ICS 111 Format: Lecture Quizzes

- Synchronous Lecture (MW 1:30-2:45) intended for interaction
- At some point during each lecture, there will be a quiz
  - each student must have a device, sufficiently charged, with internet access, to take the quiz
- Quiz will test your knowledge of the material
  - quizzes instead of exams
- Half grade given just for attendance, so turn in the quiz even if you don't know the answer
- Must be done individually!!!

Assignments

- Assignments require programming
- Programming requires a development environment
  - Integrated Development Environment, **IDE**
  - Many IDEs, for this class use **Eclipse**



```

b


HelloWorld {


 args

atic void main(String[] args) {
DO Auto-generated method stub
m.out.println ("Hello Edoardo Biagioni!");
```

0 items

Description	Resource	Path

:reate a Hello World ıpplication

- Introduction
- Open the Java perspective
- Create a Java project
- Create your HelloWorld class
- **Add a print statement**

Now that you have your HelloWorld class, in the **main()** method, add the following statement:

System.out.println("Hello world!");

Then **save** your changes; the class will automatically compile upon saving.

Click when complete

- Run your Java application

Integrated Development Environment

- Programs are written in a **programming language**
  - in this class, we use **Java**
- a **compiler** translates the program so it can be executed
- a command line lets you run the program
  - and lets you give arguments to the program
- a **debugger** helps you understand what the program is doing
- an **editor** lets you write the program
- IDE integrates these

Non-Integrated Development Environment

- Programs are written with a text editor
  - vi, nano, notepad

```
% nano Hello.java
```

- a **compiler** translates the program so it can be executed

```
% javac Hello.java
% java Hello
```

- a **debugger** helps you understand what the program is doing

Assignment 1

- Install an IDE on a device you have access to
- Write a program to print “Hello, world!”
- Compile and run
- Modify the program, to print

  “Hello, ***your name***!”.
  - ***your name*** is a placeholder for your actual name
- Compile, run, and submit
  - you get more credit if your program compiles, and full credit if it compiles and runs correctly
- it's a good idea to do this assignment completely

Errors

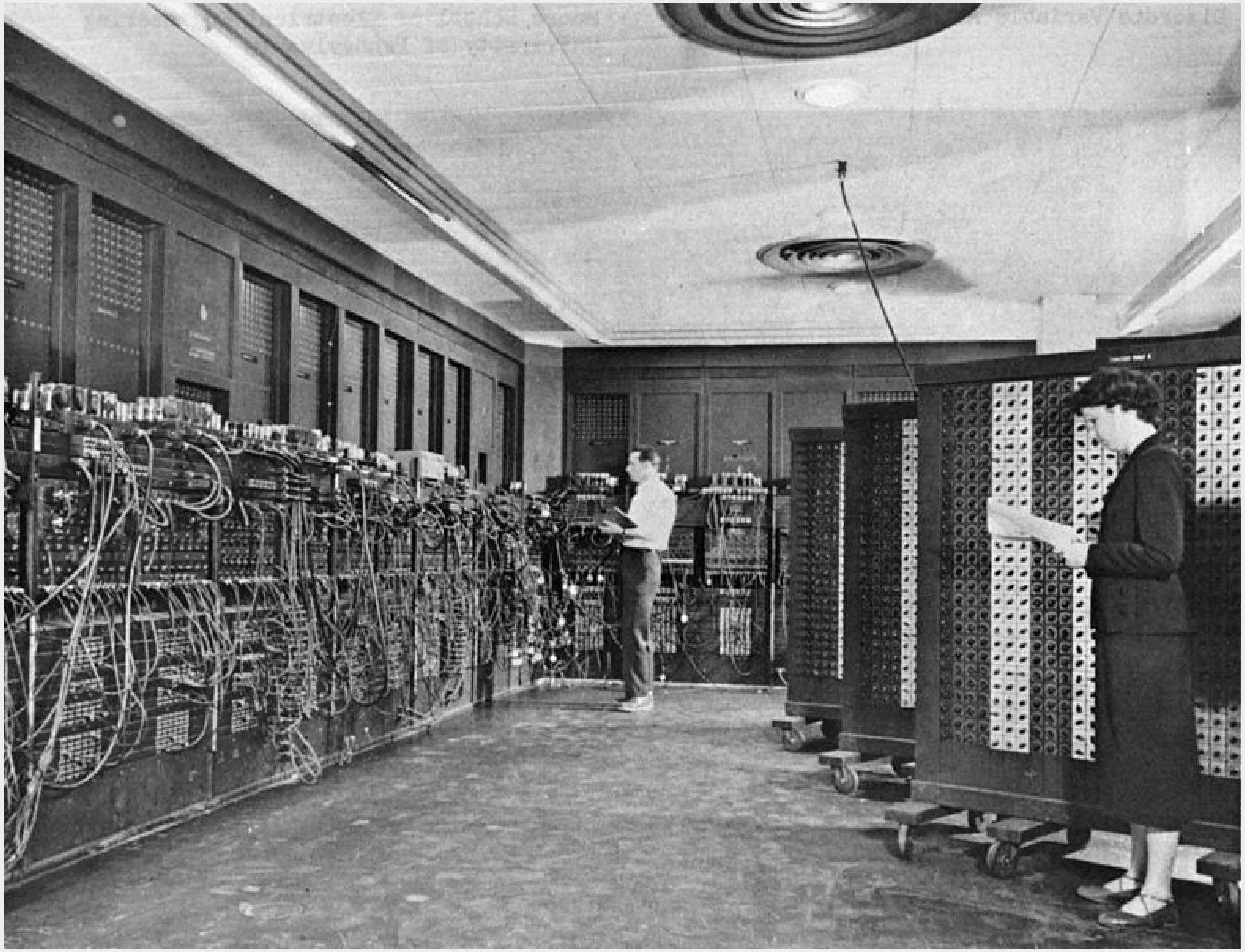
- There's a good chance your program had one or more errors
- compilation errors must be fixed before the program can run
  - e.g. if missing the semicolon “;” at the end of a statement
  - compilation errors are relatively easy to find and fix
- runtime errors show up once the program is run
  - runtime errors are sometimes easy, sometimes very hard to find and fix
  - e.g. printing the wrong string

How did we get here?

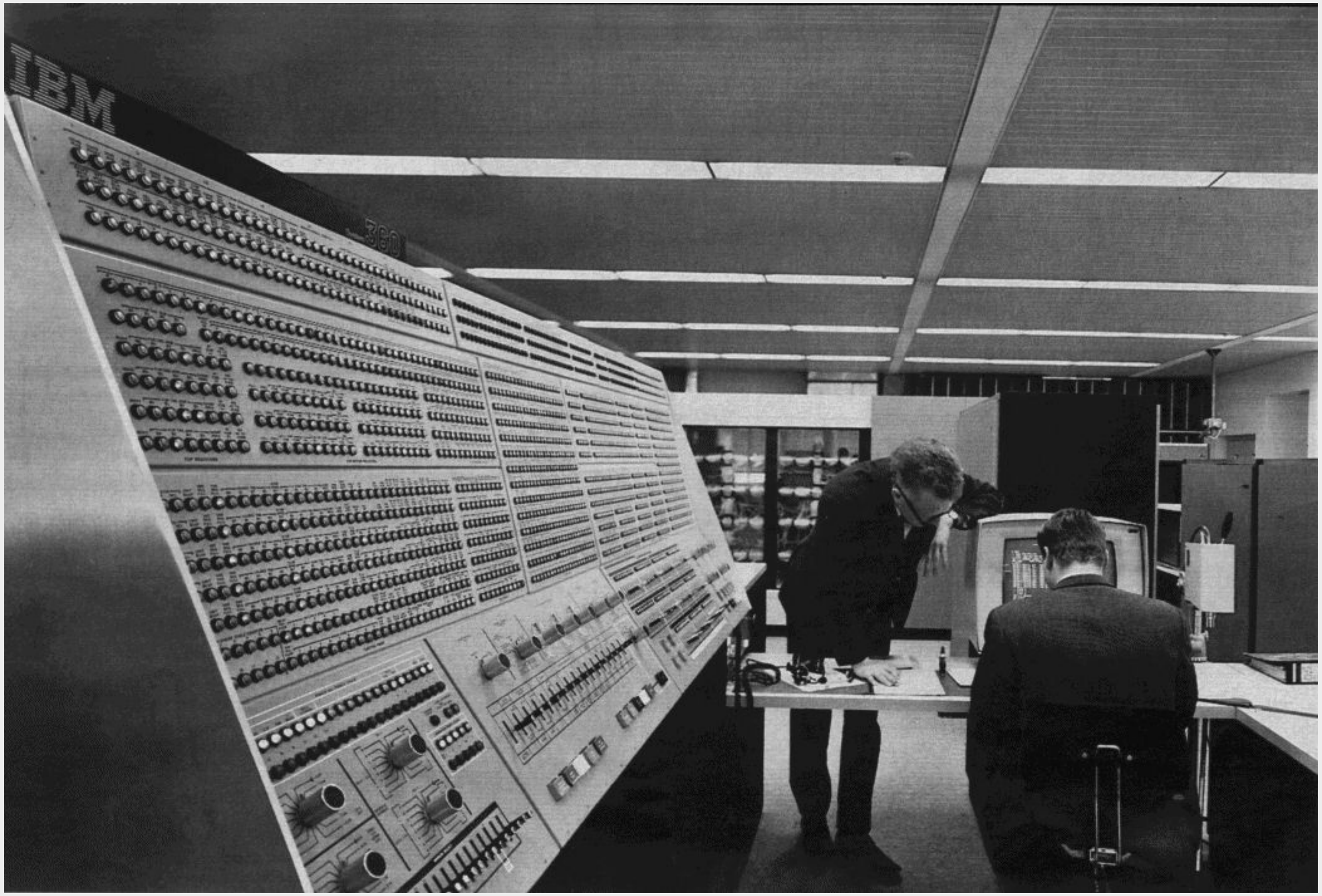
- Long history of computing
- started with special-purpose “computing machines”
- later, computers started to execute generic programs
  - a program is a collection of bits in the memory of a computer
  - it is very easy to make mistakes with collections of bits!

    01011101011010101001010111010111010010101000
- John Backus in the late 1950s developed the first high-level programming language, **Fortran**
- Programming languages make programs easier to read, understand, and maintain/change

Eniac



IBM 360



IBM PC

IBM
MS-DOS 3.1
Operating System
Microsoft
IBM's PERSONAL COMPUTER
ARC

Smartphones

Programming Language Evolution

- Fortran was very careful with storage
- 1960s-70s languages better with strings, variable allocations, recursion:
  - COBOL, PL/1, C, Pascal, APL, Algol, LISP
- 1980s and later scripting languages:
  - perl, python, ruby
- 1990s object-oriented languages:
  - Java, Javascript
- many, many, many languages!

Summary

- A program tells a computer what to do
- Writing correct programs is hard
  - Many tools help write better programs:
  - programming languages
  - development environments
- but worth it!