CSE 21
Intro to Computing II

Lecture 1 – General Course Information

Review of CSE20 (1)

CSE 21: Fall 2019

- Instructor
  - Santosh Chandrasekhar
  - schandrasekhar@ucmerced.edu
  - Office Room: AOA 143
  - Office Hours:
    - **MW 9:30-11:30am**
    - Open door policy: If you happen to pass by AOA 143 when I'm in office, please knock and come in
    - By appointment
- TAs
  - Yuan Ren, yren5@ucmerced.edu, 02L/03L
    - Office Hours: W 3:00-5:00pm, COB1 376
  - Maryam Khazaei Pool, mkhazaeipool@ucmerced.edu, 04L
    - Office Hours: W 7:30-8:30pm, SE1 388

*Syllabus updated with TA office hours and location by Aug 31st

Email Policies

- All email inquiries received before 5pm during school days will be replied within 48 hours
- I may not be able to answer emails received after 5:00pm on weekdays, or during weekend/holidays. Please plan accordingly
- IMPORTANT!
  - You **must** follow the guidelines listed in the email etiquette document available in CatCourses for ALL email communications in this course

Lecture Guidelines

- Your success is my success
  - Lectures are only successful when you understand the material being presented
- Please ask questions
  - Raise your hand, or just speak up
  - Don’t be shy, you are not being graded for raising a point or asking for clarifications
- Please bring your comments, suggestions, interesting points, or even disagreements to my attention during or after class
  - I/We can learn from you too
  - New ideas and discussions make the class much more lively and interesting

Lecture Guidelines

- Please be courteous of others.
- Examples of inappropriate or distracting classroom behavior include
  - Conversing during lecture
  - Checking E-mail or Facebook
  - Internet use not related to current class topic
  - Sleeping/Snoring?
- Turn off (or silence) your cell phones

Announcements

- Today: Introductory Material, Course Details, Review of CSE20 (1)
- Lab
  - Lab 1 (Knowledge Check) assigned next week (09/01 – 09/07)
    - Due in one week (plus additional **3 days** grace period)
    - Make sure to demo your work to YOUR OWN TA (or me) after submission
    - Demo is REQUIRED to receive credit for assignment
- Reading Assignment
  - Review all Sections in Chapters 1 – 5 covered by CSE20 (**Not graded**)
  - Reading 01 (6.1 – 6.11) due 09/11
    - Complete Participation Activities in each section to receive grade towards Participation
    - IMPORTANT: Make sure to **submit score to CatCourses** by using the link provided on CatCourses

Course Overview

- Prerequisites
  - CSE20
  - Basic knowledge of Computer Science
- CatCourses
  - Check regularly for announcements.
  - Labs & Project Assignments will be posted and submitted there.
  - Grades for assignments will also be found there (secure).
- 1 Lecture and 1 Lab per week
- 1 Midterm exam (Oct 16, tentative)
- Final exam (Dec 14)
- 14 lab assignments (every week)
  - Includes two exam review labs
- 2 programming projects

Course Overview

- Grading policy
  - Participation: 15%
    - Lab Participation + zyBook Participation Activities
  - Projects: 15%
  - Mid-term: 20%
  - Lab assignments: 25%
  - Final exam (comprehensive): 25%

- Grading scheme (tentative):
  - ≥ 88% scores at least an A-
  - ≥ 73% scores at least a B-
  - ≥ 59.5% scores at least a C-

Course Material

- Textbook:
  - Programming in Java by ZyBooks
    - Sign up at zybooks.com using your UC Merced Email ID
    - Enter zyBook code: **UCMERCEDCSE021ChandrasekharFall2019** (case sensitive)
    - Subscribe
  - **You must subscribe your own copy with your UCM email.**
    - Participation grade will be evaluated based on the activities within the subscription account.
  - See syllabus for details about potential discounts

Lab Rules

- Attendance is mandatory
  - Participation grade is evaluated based on physical presence during lab hours. **> 80% attendance** to receive full lab participation grade, or none otherwise.
  - Your TA will take attendance during lab hours
- Submit on CatCourses before the deadline
  - Grace period of THREE days after deadline with no penalty.
  - No submission after grace period (exceptions may apply if approved by instructor **beforehand**).
- To ensure that your assignments receive a grade, you MUST **show/demo** your work to your TA or instructor **before submission period ends**, and we will ask you questions related to your work.
  - **Demo MUST be done AFTER submission but BEFORE end of grace period.**
  - **Submissions without a demonstration will receive a grade of ZERO.**

Project Rules

- 1 – 2 students per group.
- All group members must submit their own solution in their CatCources account.
- Should be done outside of lab session hours unless you have completed the lab assignment already.
- Same submission policy as labs, except for later deadline (typically 2 weeks)

Code Demo

- **Be prepared**
  - May be asked to perform a walkthrough of code that involves presenting the program in a step-by-step manner to the TA or instructor and answering any assignment-related questions that are posed
  - Questions about lab assignments or projects can be wide-ranging. Eg.,
    - Explain portions of code in detail,
    - Provide reasons behind decisions and choices,
    - Predict program behavior when modifications are introduced, etc.
  - Questions will be used to test knowledge of programming and theoretical concepts relevant to lab assignment or project being evaluated

Code Demo

- You can demo your code to ANY TA (preferably your own) during the following times:
  - Lab hours (found in UC Merced Schedule)
  - Office hours (found in syllabus)
  - By appointment
- If no TA is available (or on direction by a TA), you may demo your code during instructor's office hours

Exams

- 45% of the course grade
  - Midterm 20%
  - Final 25%
- Open notes and open book (chapter printouts)
  - Just don't bring in electronic devices
- Practice Exams
  - For both midterm and final
  - Actual exam will follow a similar format and order
  - Expect you to study hard so each problem will be harder on the actual exam

Policies

- Don’t copy someone else’s code
- Don’t give your code away
- Don’t outsource your assignments
- Don’t use electronic devices in exams
- Don’t use electronic devices during lecture for purposes other than note taking
- Turn off speakers/cellphone during class
- No Cheating!
  - Communicating information to another student during examination.
  - Knowingly allowing another student to copy one’s work.
  - Offering another person’s work as one’s own.
  - I am serious!

Academic Integrity

- Academic integrity is the foundation of an academic community. Academic integrity applies to research as well as undergraduate and graduate coursework.
- Academic misconduct includes, but is not limited to cheating, fabrication, plagiarism, altering graded examinations for additional credit, having another person take an examination for you, or facilitating academic dishonesty or as further specified in this policy or other campus regulations.
- For more information, please visit http://studentconduct.ucmerced.edu/ and review the following documents
  - UCM Code of Student Conduct
  - UCM Academic Honesty Policy

Academic Integrity

- **Cheating** is the unauthorized use of information in any academic exercise, or other attempts to obtain credit for work or a more positive academic evaluation of work through deception or dishonesty.
- **Plagiarism** refers to the use of another's ideas or words without proper attribution, or credit.
- **Collusion** is when any student knowingly or intentionally helps another student to perform any of the above acts of cheating or plagiarism.
  - Students who collude are subject to discipline for academic dishonesty.
  - No distinction is made between those who cheat or plagiarize and those who willingly facilitate cheating or plagiarism

Academic Integrity

- Please note that with respect to the programming assignments we run your code through a system to detect similarity with other projects submitted by your classmates, online solutions, and a database of previous year's submissions.
- The algorithm analyses the structure and flow of the code, so simply changing the variable names and introducing minor changes will not be effective to defeat it.
- You would have to modify the code so much, that you are better off writing the code on your own.
- So **caveat emptor**.

Academic Integrity

- **Cheating vs. Collaboration**: So how do you draw the line between collaboration and cheating? Here's a reasonable set of ground rules. Failure to understand and follow these rules will constitute cheating and will be dealt with as per university guidelines.
  - **The Simpson's Rule**: This rule says that you are free to meet with fellow student(s) and discuss assignments with them. Writing on a board or shared piece of paper is acceptable during the meeting; however, you should not take any written (electronic or otherwise) record away from the meeting. This applies when the assignment is supposed to be an individual effort or whenever two teams discuss common problems they are each encountering (inter-group collaboration). After the meeting, engage in a half hour of mind-numbing activity (like watching an episode of the Simpsons), before starting to work on the assignment. This will assure that you are able to reconstruct what you learned from the meeting, by yourself, using your own brain.
  - **The Freedom of Information Rule**: To assure that all collaboration is on the level, you must always write the name(s) of your collaborators on your assignment.

Computer Science Department Policy on Academic Honesty

- Effective Fall 2019: how Academic Honesty Policy (AHP) will be implemented when students enrolled in classes offered by CSE
- For more details, see Computer Science Department Policy On Academic Honest - Fall 2019
- First Infraction
  - Student receives a 0 (or equivalent grade) for the assignment
  - Reported to the Dean of the School of Engineering
  - Reported to Office of Student Conduct for review of possible violations of the Code of Student Conduct
- Additional Infractions
  - If one or more prior violations of the AHP (**in the same or another course**), student receives a failing grade (F) for the course WITHOUT THE ABILITY TO WITHDRAW FROM CLASS

Hints for success

- Attend lecture
- Read the textbook and work on the activities
- Do & understand the assignments YOURSELF
- Create a portfolio to save all your work
- Take notes while reading and in lecture
- Ask questions: We are here to help you!
- Helpful resources posted on CatCourses
- Enjoy Learning!

Welcome to the Class!

Review of CSE 20

- Problem Statement
  - We want to survey the type of mobile OS students prefer. It will ask for a sample size and inquire for each person whether they like Android, iOS or both. It should report a breakdown of the data upon request.

The Program Skeleton

```
public class PreferenceMOS {

    public static void main(String[] args) {



    }

}
```

What's the first thing?

- Get input from user
  - How? Use a Scanner

Scanner

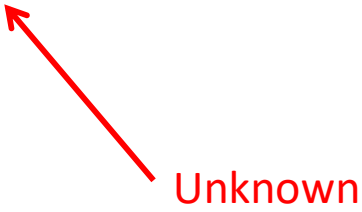
```
public class PreferenceMOS {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

    }

}
```



Import Class

```
import java.util.Scanner;

public class PreferenceMOS {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

    }

}
```

Steps

- Get input from user
  - How? Use a Scanner
  - What? To begin with, the sample size

Get sample size

```
import java.util.Scanner;
public class PreferenceMOS {
        public static void main(String[] args) {
                Scanner input = new Scanner(System.in);
                System.out.print("Enter the total number of students: ");
                int max = input.nextInt();
        }
}
```

Steps

- Get input from user
  - How? Use a Scanner
  - What? To begin with, the sample size
- Get samples

Get samples

- Ask to choose which one they prefer
  - Print
    - 1 for Android
    - 2 for iOS
    - 3 for Both
    - 4 for Other
- Use tally counters
  - if choice is
    - 1, android++
    - 2, ios++
    - 3, android++, ios++

Code to get a sample

```
System.out.println("Preference? Android (1), iOS (2), Both (3), or Other (4).");
System.out.print("Enter choice: ");
int choice = input.nextInt();

if (choice == 1)
        android++;
else if (choice == 2)
        ios++;
else if (choice == 3) {
        android++;
        ios++;
} else if (choice == 4)
        other++;
else
        System.out.println("Invalid choice.");
```

Putting it all together

```
import java.util.Scanner;

public class PreferenceMOS {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        int android, ios, other;
        android = ios = other = 0;

        System.out.print("Enter the total number of students: ");
        int max = input.nextInt();

        System.out.println("Preference? Android (1), iOS (2), Both (3), or Other (4).");
        System.out.print("Enter choice: ");
         int choice = input.nextInt();`
        if (choice == 1) android++;
        else if (choice == 2) ios++;
        else if (choice == 3) {
                android++;
                ios++;
        } else if (choice == 4) other++;
        else System.out.println("Invalid choice.");

    }
}
```

Initialize all counters

Sample size

Get a sample

Steps

- Get input from user
  - How? Use a Scanner
  - What? To begin with, the sample size
- Get samples
  - For each person
    - Ask for choice (gather information): 1 for Android, 2 for iOS, 3 for both (any other input tallied under "other").
    - Use tally counters:
      If choice is 1, android++
      2, ios++
      3, android++ and ios++

Repeat for each student

```
System.out.print("Enter choice: ");
choice = input.nextInt();
if (choice == 1) android++;
else if (choice == 2) ios++;
else if (choice == 3) {
        android++;
        ios++;
} else if (choice == 4) other++;
else System.out.println("Invalid choice.");

System.out.print("Enter choice: ");
choice = input.nextInt();
if (choice == 1) android++;
else if (choice == 2) ios++;
else if (choice == 3) {
        android++;
        ios++;
} else if (choice == 4) other++;
else System.out.println("Invalid choice.");
```

```
System.out.print("Enter choice: ");
choice = input.nextInt();
if (choice == 1) android++;
else if (choice == 2) ios++;
else if (choice == 3) {
        android++;
        ios++;
} else if (choice == 4) other++;
else System.out.println("Invalid choice.");

System.out.print("Enter choice: ");
choice = input.nextInt();
if (choice == 1) android++;
else if (choice == 2) ios++;
else if (choice == 3) {
        android++;
        ios++;
} else if (choice == 4) other++;
else System.out.println("Invalid choice.");
```

Too many people?

Looping

```
int choice = 0;
for (int i = 0; i < max; i++) {
        System.out.print("Enter choice: ");
        choice = input.nextInt();
        if (choice == 1)
                android++;
        else if (choice == 2)
                ios++;
        else if (choice == 3) {
                android++;
                ios++;
        } else
                other++;
}
```

Ignore invalid choices, and assume any other input is tallied under "other"

Steps

- Get input from user
  - How? Use a Scanner
  - What? To begin with, the sample size
- Get samples
  - For each person (for loop)
    - Ask for choice (gather information): 1 for Android, 2 for iOS, 3 for both (any other input tallied under "other").
    - Use tally counters:
      If choice is 1, android++
      2, ios++
      3, android++ and ios++
- Output

Output

```
System.out.print("See detailed count? Yes (1), or No (0): ");
int detailed = input.nextInt();
if (detailed == 1) {
        System.out.println("Prefer Android = " + android);
        System.out.println("Prefer iOS = " + ios);
        System.out.println("Prefer Other = " + other);
}
```

Final Code

```
import java.util.Scanner;

public class PreferenceMOS {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter the total number of students: ");
        int max = input.nextInt();
        int android, ios, other, choice;
        android = ios = other = choice = 0;
        System.out.println("Preference? Android (1), iOS (2), or Both (3).");

        for (int i = 0; i < max; i++) {
            System.out.print("Enter choice: ");
            choice = input.nextInt();
            if (choice == 1) android++;
            else if (choice == 2) ios++;
            else if (choice == 3) {
                android++;
                ios++;
            } else other++;
        }

        System.out.print("See detailed count? Yes (1), or No (0): ");
        int detailed = input.nextInt();
        if (detailed == 1) {
            System.out.println("Prefer Android = " + android);
            System.out.println("Prefer iOS = " + ios);
            System.out.println("Prefer Other = " + other);
        }
    }
}
```