First Recitation for Data Structures

- [ ] 1. Introduction + Resources.
  - [ ] office hour & contact
  - [ ] Motivation
  - [ ] Advice to learn this class
  - [ ] what I will do in recitation
  - [ ] reminder for reading list
- [ ] 2. IDE
- [ ] 3. Object-oriented Programming (OOP)
  - [ ] Motivation
  - [ ] Inheritance
  - [ ] Interfaces and Abstract classes
  - [ ] Exceptions
- [ ] 4. Coding demo.

1. Introduction + Resources

- Office Hour & Contact

  Thursday 11-13 60 Fifth Avenue. 350

  charlie.chen@nyu.edu

- Motivation

  Programming = Algorithm + Data Structure

  Job interview.

- Advice for this class
  1. balance thinking and asking/discussing.
  2. visualize.
  3. practice (write codes)

- what I will do in recitation.

  Half teaching + half coding.

  Examples.

- Reminder for reading list

  https://github.com/AlanNawzadAmin/CSCI-UA-201-011-Spring-2024/tree/main

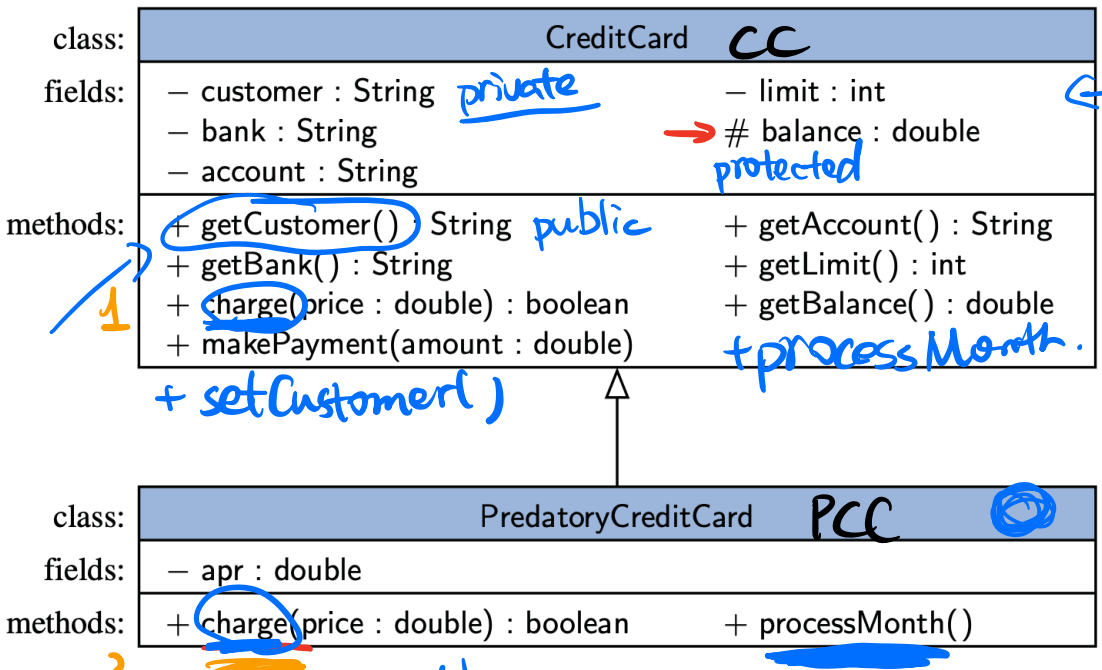
2. IDE

3 components.

3. Object-Oriented Programming (OOP)

- Motivation
  - Modularity & Reusability.
  - Encapsulation & Data protection.
  - Abstraction & code mangement

- Inheritance



private
protected
public
protected idx;
+ processMonth.
+ setCustomer()
1
2.

pcc.idx.
pcc.getCustomer()
pcc.customer
pcc.setCustomer()

Access modifier { public, private, protected. }

polymorphism

dynamic dispatch

Code	Expected behavior
CC card = new PCC();	Correct.
card.getCustomer();	correct.
card.charge(2.0);	correct
card.processMonth();	incorrect.

- Interfaces & Abstractions

- Interface

```
1  /** Interface for objects that can be sold. */
2  public interface Sellable {
3
4    /** Returns a description of the object. */
5    public String description();
6
7    /** Returns the list price in cents. */
8    public int listPrice();
9
10   /** Returns the lowest price in cents we will accept. */
11   public int lowestPrice();
12 }
```

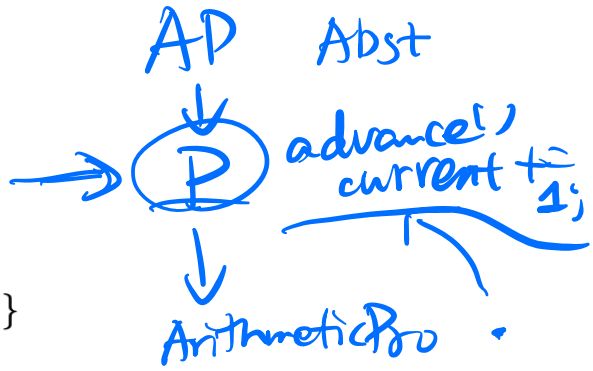
Code Fragment 2.8: Interface Sellable.

Sellable Photograph;

```
1  /** Class for photographs that can be sold. */
2  public class Photograph implements Sellable {
3    private String descript;                    // description of this photo
4    private int price;                          // the price we are setting
5    private boolean color;                      // true if photo is in color
6
7    public Photograph(String desc, int p, boolean c) {    // constructor
8      descript = desc;
9      price = p;
10     color = c;
11   }
12
13   public String description() { return descript; }
14   public int listPrice() { return price; }
15   public int lowestPrice() { return price/2; }
16   public boolean isColor() { return color; }
17 }
```

Code Fragment 2.9: Class Photograph implementing the Sellable interface.

– Abstraction




```
1  public abstract class AbstractProgression {
2    protected long current;
3    public AbstractProgression( ) { this(0); }
4    public AbstractProgression(long start) { current = start; }
5
6    public long nextValue( ) {                  // this is a concrete method
7      long answer = current;
8      advance( );     // this protected call is responsible for advancing the current value
9      return answer;
10   }
11
12   public void printProgression(int n) {       // this is a concrete method
13     System.out.print(nextValue( ));           // print first value without leading space
14     for (int j=1; j < n; j++)
15       System.out.print(" " + nextValue( )); // print leading space before others
16     System.out.println( );                    // end the line
17   }
18
19   protected abstract void advance( );         // notice the lack of a method body
20 }
```

Code Fragment 2.12: An abstract version of the progression base class, originally given in Code Fragment 2.2. (We omit documentation for brevity.)

- Exception

```
try {
    guardedBody
} catch (exceptionType_1 variable_1) {
    remedyBody_1
} catch (exceptionType_2 variable_2) {
    remedyBody_2
} ...
    ...
```

```
1  public static void main(String[ ] args) {
2    int n = DEFAULT;
3    try {
4      n = Integer.parseInt(args[0]);
5      if (n <= 0) {
6        System.out.println("n must be positive. Using default.");
7        n = DEFAULT;
8      }
9    } catch (ArrayIndexOutOfBoundsException e) {
10     System.out.println("No argument specified for n. Using default.");
11   } catch (NumberFormatException e) {
12     System.out.println("Invalid integer argument. Using default.");
13   }
14 }
```

Code Fragment 2.13: A demonstration of catching an exception.

4. Code demo

- Polymorphism
- Multiple dispatch.
- access modifier.
- abstract class.