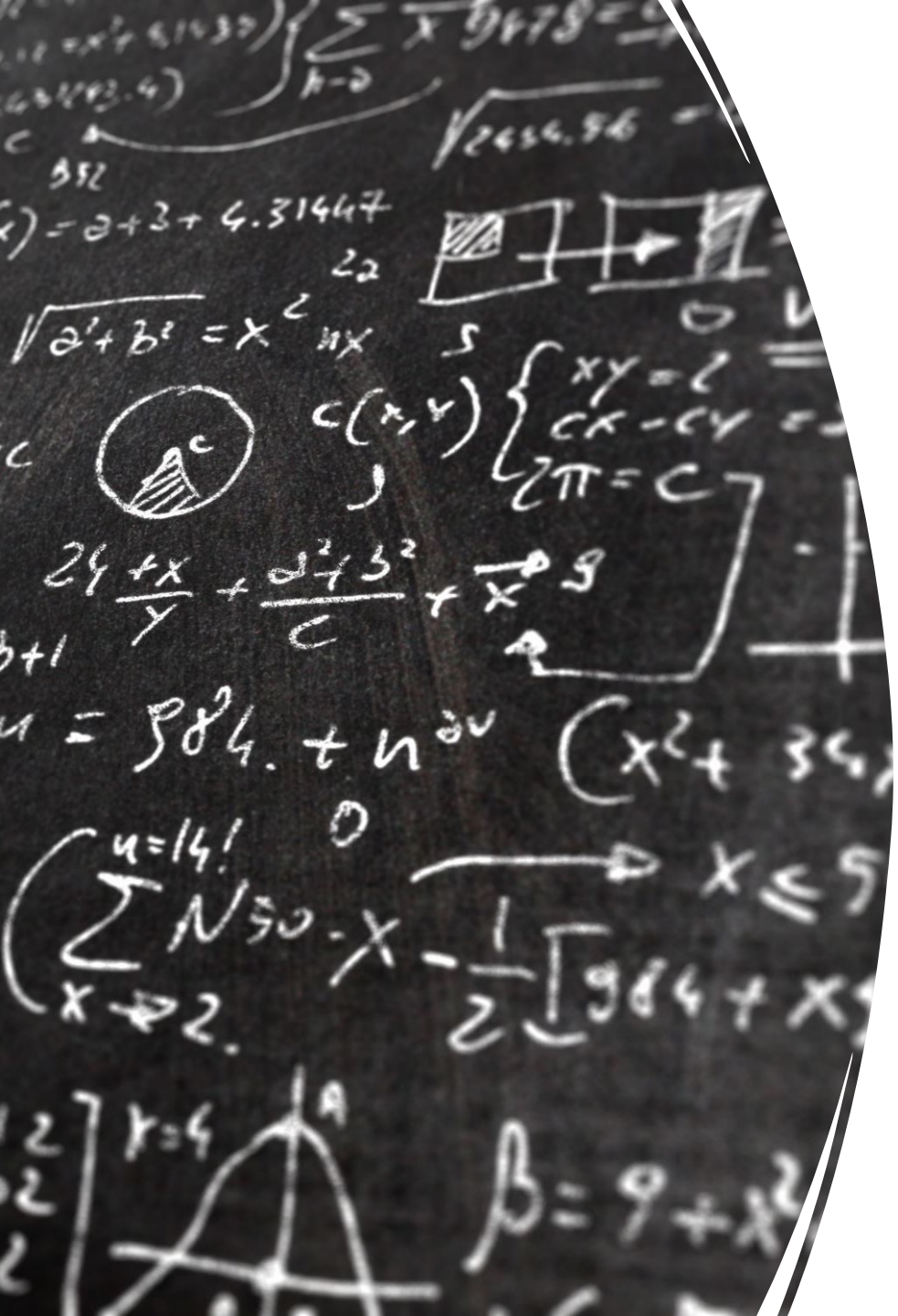


CS 113 – Computer Science I

Lecture 15 – Classes I

Adam Poliak

03/14/2023

Announcements

- HW06
  - Due Monday 03/20
- Midterms:
  - Grades by end of this week
  - Was sick over Spring break
- Week after spring break
  - Hell week – first ten minutes of Thursdays class
- Midsemester feedback

Agenda

- Reading files
- Mutability
- Objects

Reading files

Use Scanner class

File class

```
File input = new File("file.txt");
Scanner sc = new Scanner(input);
System.out.println(sc.nextLine());
```

Errors

Need to either:

- indicate that the method can cause the specific `FileNotFoundException`
- Or write code to `catch` it
  - Deal with the case where the specified file is not found

More in lab

Back at our ArrayUtilities

```java
public class ArrayTest {

  public static void add1(int[] numbs, int index) {
    if (index == numbs.length) {
      return;
    }
    numbs[index]++;
    add1(numbs, index+1);
  }

  public static void main(String[] arg) {
    int[] numbs = {1, 2, 3, 4};

    for (int numb : numbs) {
      System.out.print(numb + " ");
    }
    add1(numbs, 0);
    for (int numb : numbs) {
      System.out.print(numb + " ");
    }

  }

}
```

Adding 1 to each value in the array

What is the numbs at the end of this?

This is called *modifying in place*

Agenda

- Reading files
- **Mutability**
- Objects

Mutable vs Immutable

Mutable:

Values can change

Immutable:

Values cannot change

Strings and Integers are immutable

Agenda

- Reading files
- Mutability
- **Objects & Classes**

Data types revisited

What are some examples of built-in types in Java?

What is a data type?

Examples

Type	Valid values	Operations

Examples

Type	Valid values	Operations
int	1, 10, 999	%, +, -, / …
boolean	true, false	==, &&, ||, !=
String	Anything between “”	.compareTo(), .charAt(), concatentation, …

Class

A blueprint for a custom data type

A template for how data/information is stored

Contains a set of methods for how to interact/operate on the stored data

Classes and objects

An **object** is an ***instance*** of a **class**

An **object** is to a **class** as a

cat is to an **animal**

tulip is to an **flower**

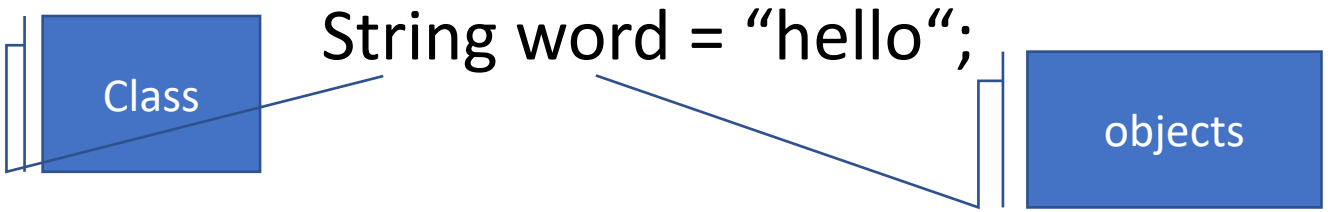
cookie it to a **snack**

Socrates is to a **human**

Classes and objects

A **class** defines the characteristics of a type (data and methods)

An **object** is a particular example of a class



Java is a strict object-oriented programming language, meaning all code must be inside a class!

Creating objects

Declare variables in the same way!

Create using `new`

Using objects

The methods you are allowed to call on an object is called an **API**

Recall: API = Application Programming Interface

Example: The *String API* has over 60 methods!

Objects can have either *static* or *instance* methods

static methods use syntax <ClassName>.<methodName>

instance methods use syntax <object>.<methodName>

Example: String API

boolean	**endsWith**(String suffix) Tests if this string ends with the specified suffix.
boolean	**equals**(Object anObject) Compares this string to the specified object.
boolean	**equalsIgnoreCase**(String anotherString) Compares this String to another String, ignoring case considerations.
static String	**format**(Locale l, String format, Object... args) Returns a formatted string using the specified locale, format string, and arguments.
static String	**format**(String format, Object... args) Returns a formatted string using the specified format string and arguments.

Using objects: some special methods

The **constructor method** is called when you do a `new`

accesors (aka getters)
return the values of instance variables

mutators (aka setters)
set the values of instance variables

toString()
returns a string representation of an object

Defining classes

By defining our own classes, we can create our own data types

A class definition contains

- the data contained by the new type (**instance variables**)

- the operations supported by the new type (**instance methods**)

Example: Defining a class `Point`

What data should it have?

What operations should it support?

Object-oriented programming (OOP)

Method for designing programs in terms of objects

Recall: Top-down design

- the “nouns” in your feature list correspond to classes/data
- the “verbs” correspond to methods

OOP Example & Design: Bank

OOP Design: Bank

Defining the Bank class

```
public class Bank {
    private int size;
    private String name;
    private String[] clients;
    private double[] accounts;

    public Snack(String bankName, int numClients) {
        name = bankName;
        size = numClients;
        clients = new String[size];
        accounts = new double[size];

    }
    public String getName() {
        return name;
    }

}
```

Testing the Bank class

```
public static void main(String args[])
{
    Bank boa = new Bank(”Bank of America", 10);
    System.out.println(”Bank: "+boa.getName());
}
```

Objects: Stack diagrams revisited

```
public static void main(String args[])
{
    Bank boa = new Bank(”Bank of America", 10); //call constructor
    System.out.println(”Bank: "+boa.getName());
}
```

Exercise: draw a stack diagram for this program

```
public static void main(String args[])
{
    Bank boa = new Bank(”Bank of America", 10); //call constructor
    System.out.println(”Bank: "+boa.getName());
}
```

Exercise: Define a class BankAccount

BankAccount should have the following data:

- Name
- Amount

BankAccount should have the following operations:

- currentBalance() // returns current amount in the bank account
- withdraw(float amt) // withdraw the given amount from the account
- deposit(float amt) // deposit the given amount to the account