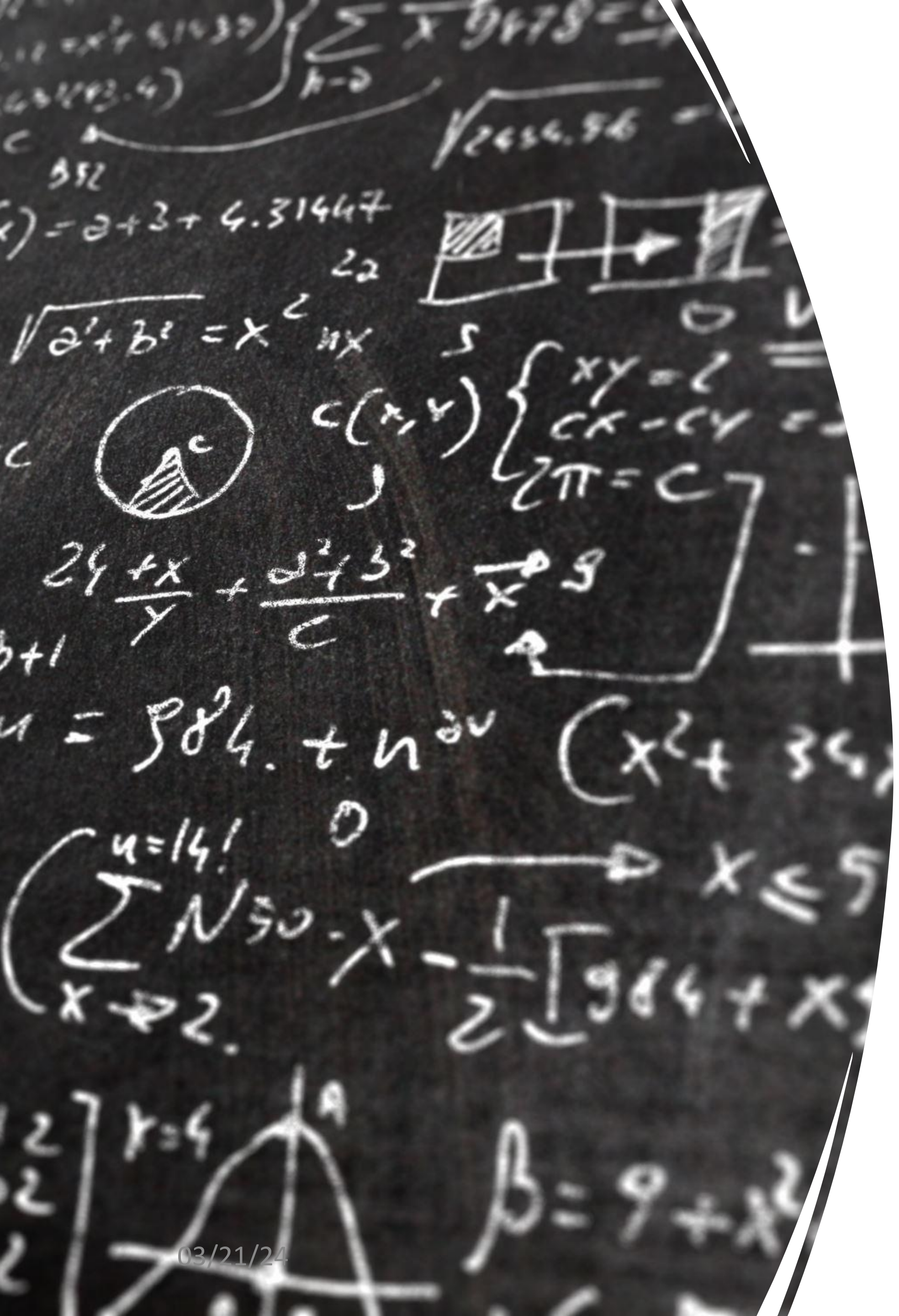

# CS 113 – Computer Science I

# Lecture 15 – Classes & Objects

Thursday 03/21/2024

# Announcements

HW 07 Released tonight

Due March 28th (next thursday)

Anonymous Course survey - 5 bonus points on your exam:
https://forms.gle/RytfNAYRUMQk24M86

Email to let me know you’ve completed it.

# Classes and Objects

# 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 | | |
| | | |
| | | |

# Examples

| Type | Valid values | Operations |
|---|---|---|
| int | 1, 10, 999 | %, +, -, / … |
| | | |
| | | |

# Examples

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

# Creating our own data type!

Rectangle data type

- setHeight
- setWidth
- getArea
- getParameter
- ...

# Creating our own data type!

Your turn... Name some ideas for a data type an operations we might want to perform on them

# 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

Let’s code a mini Rectangle class

# Classes and objects

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

A concrete occurrence of an object at runtime

```
Scanner sc = new Scanner(System.in);
```

`Scanner` is the class, `sc` is the instance.

# Classes and objects

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

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

```
String word = “hello“;
```

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`

```
Scanner sc = new Scanner(System.in);
```

# Constructors

- Special method with same name as the class

- Initializes the newly created object

- let’s write one!

# Getters

- also called accessors

- return the value of instance variables

- usually named `get`**`Var`** where **Var** is the name of the variable we want to access

# Setters

- also called modifiers

- changes the value of instance variables

- usually named `set`**`Var`** where **Var** is the name of the variable we want to modify

# toString

- returns a string representation of the object

# Summary: special methods

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

**Getters (aka accessors)**
return the values of instance variables

**Setters (aka modifiers)**
set the values of instance variables

**toString()**
returns a string representation of an object

# Static vs Non Static

# Static vs Non Static

**Static:**

Belongs to the class rather than to any particular instance of the class

Can be invoked without the need for creating an instance of the class

**Non Static (instance):**

Belongs to the object (instance) of the class

Can be invoked only through an instance of the class.

# Static vs Non Static

- Let’s make a static method for rectangles

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

static methods use syntax <ClassName>.<methodName>

instance methods use syntax <object>.<methodName>

# In Summary: 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**)

# OOP Design: Bank

Let’s create a Bank class.

1. What does a bank have? What member variables should it hold?

2. How should those values be initialized?

3. What actions should we be able to perform on bank?
    a. How can we find out how much money the bank is holding at once?
    b. How can we find out which account is currently overdraft?
    c. What other questions might the bank want to know?

# 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

# 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