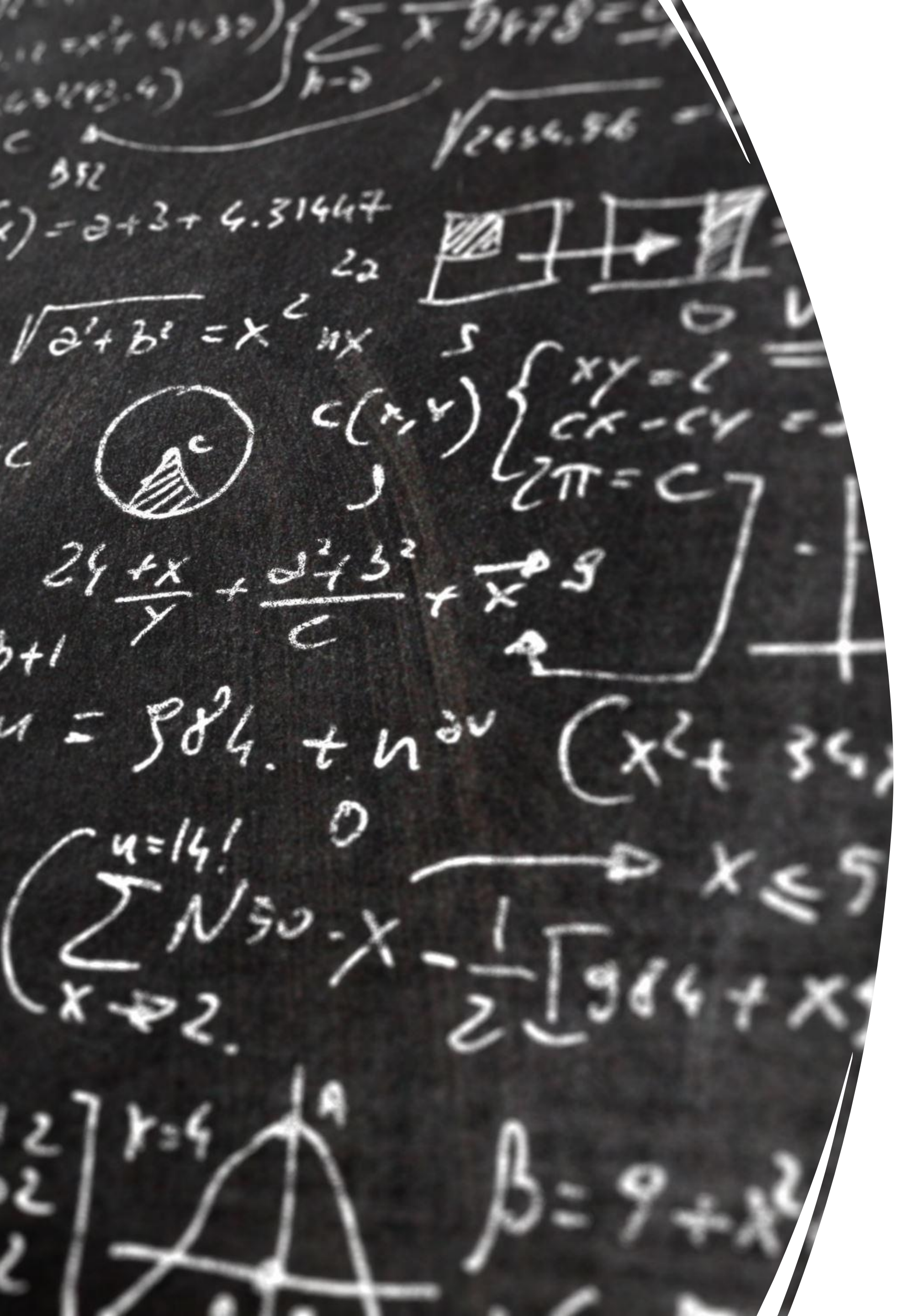

CS 113 – Computer Science I

Lecture 18 – Relationships & Class Actions

Tuesday 4/2/2024

Announcements

HW06 & Lab06 due Thursday (4/4)

Lab06 checked off by TAs

Outline

- Inheritance Review
- Interfaces

Class Review

1. What is a class?

2. What is a constructor?
    a. When is it executed?
    b. What code is typically in a constructor?
    c. What are two types of constructors?

3. What are getters and setters?
    a. Why do we need them?

4. What is toString()

Another special class method

`equals(Object o)`

What happens if you check equality of two objects using == ?

How should we compare two strings?

How should we compare two BankAccounts?

- write our own `equals`!

Rules:

1. Reflexivity: An object should always be equal to itself
2. Consistency: Calling equals() on the same objects should return the same thing each call

More class review

- this keyword
  - What does it mean?
  - When should we use it?

- What are access modifiers
  - public?
  - private?
  - protected?
  - When should we use each modifier?

More class review - **Inheritance**

Inheritance

- What is it?
- What keyword do we use the inherit from a parent class?
- can we inherit from more than one parent?

- **Example:** BrynMawr Student Database
  - Each student has a name and a student ID
  - A CS student has a name, a student ID, and a goldengate username
  - A physics student has a name, a student ID, and a lab section
  - What should the hierarchy look like?
  - Let’s code it!

More class review

Hierarchy.java

polymorphism

- what is it?

super

- what is it?

Method overriding

- what is it?

Interfaces

Interfaces

- An interface is a contract - A set of shared methods that users **must** implement
- create a program to calculate the area of different shapes, such as circles, rectangles, triangles etc.
- For each shape, you should be able to print the shape name and area
- Every time someone adds a new shape, they **must** include the methods for getName() and getArea()

Interfaces

- For any new shape that is created, we want to **enforce** that these methods are also implemented.

```
interface Shape {
    public double getArea();
    public String getName();
}
```

```
class Circle implements Shape {
```

Interfaces

A contract - A set of shared methods that users **must** implement

A collection of method signatures with no bodies

A class can implement more than one interface

Interfaces

An interface is not a class!

A class is what an object **is**

An interface is what an object **does**

- can not be instantiated
- no constructors
- incomplete methods

Interface

No modifier - implicitly `public`

No instance variables except for constants (`static final`)

Inheritance vs Interfaces

Each of these lines is related to either interfaces or inheritance...

- `extends` keyword
- guarantees a class has implemented certain methods
- `implements` keyword
- reuses implementations
- *is-a* relationship
- specifies what a class *does*