# CS 152
# Computer Programming
# Fundamentals
# Classes and Objects

Brooke Chenoweth

University of New Mexico

Spring 2025

# What is an object?

An object encapsulates *data* and *behaviour*.

- Data
- State
- Properties

In Java: fields, aka member variables

- Behaviour
- Actions
- Activities

In Java: methods

# What is an object?

Each object has certain data and behavior

- An example: *student*
  - Data: age, endurance, intelligence, ...
  - Behavior: code, drink, workout, sleep, ...
- Another example: *car*
  - Data: power, top-speed, shape, color, etc...
  - Behavior: start, accelerate, break, turn

# What is a class?

- A class is a blueprint from which objects are created.
- An object created from a class is an *instance* of that class.

# Constructors

- A *constructor* is a special kind of method used to construct an instance of a class.
- Constructor name is same as class name.
- No return type (not even void!)
- Can be overloaded, like other methods.
- When creating a object by calling a constructor, use the keyword `new`

# Keyword: `this`

- Access shadowed member variables.
- Call one constructor from another.

```
public class Point2D {
  private double x, y;
  public Point2D(double x, double y) {
    this.x = x;
    this.y = y;
  }
  public Point2D() {
    this(0, 0);
  }
}
```

# Example class

```
public class Student {
  private int age, endurance, intelligence;

  public Student ( int age, int endurance, int intelligence ) {
    this.age = age;
    this.endurance = endurance;
    this.intelligence = intelligence;
  }

  public void drink ( String what ) {
    if ( what == "milk" ) {
      endurance++;
    } else if ( what == "alcohol" ) {
      if (age >= 21) {
        intelligence = intelligence - 5;
      } else {
        System.out.println("You are too young to drink!");
      }
    } else {
      System.out.println("Don't drink " + what + "!");
    }
  }
}
```

# Find mistakes!

- What's wrong with the program on previous page?

# The `String` trap

- Why can't you compare two strings with the == operator?
- Reference types!
  - A reference to a place in memory - a comparison with the == operator compares addresses of memory.
  - Are the two references both refering to the same object?
- When comparing two objects, usually want to use `equals` method.

# Example class revisited

```
public class Student {
  private int age, endurance, intelligence;

  public Student ( int age, int endurance, int intelligence ) {
    this.age = age;
    this.endurance = endurance;
    this.intelligence = intelligence;
  }

  public void drink ( String what ) {
    if ( what.equals("milk") ) {
      endurance++;
    } else if ( what.equals("alcohol") ) {
      if (age >= 21) {
        intelligence = intelligence - 5;
      } else {
        System.out.println("You are too young to drink!");
      }
    } else {
      System.out.println("Don't drink " + what + "!");
    }
  }
}
```

# Class vs Instance variables

Instance variables

- Non-static fields
- Every object has its own
- Need instance to use

Class Variables

- Static fields
- Associated with *class*, not a particular object
- Can be manipulated without an instance

# Class and Instance Variable Example

```
public class Student {
   // These are instance variables
   private String name;
   private int id;

   // This is a class variable
   private static int numberOfStudents = 0;

   public Student ( String name ) {
      this.name = name;
      // Give each student a unique ID
      this.id = ++numberOfStudents;
   }

   // More methods here...
}
```

# Access Modifiers

| | |
|---|---|
| `public` | Accessible to all |
| `private` | Only this class |
| `protected` | Only this class and its subclasses |
| *package-private* | No modifier. This class and others in same package. |

(For CS152, we'll only be using `public` and `private`, but you may see the other modifiers in your textbook or in example code on the internet.)

# Access Modifier Tips

- Don't expose your guts!
- Use `private` unless you have a good reason not to.
- Avoid public fields except for constants. (Use getter/setter)