

More Classes



Colorado State University
Department of Computer Science

Slides Originally Created by Albert Lionelle (Albert.Lionelle@colostate.edu)
updated by Marcia Moraes (marcia.moraes@colostate.edu)

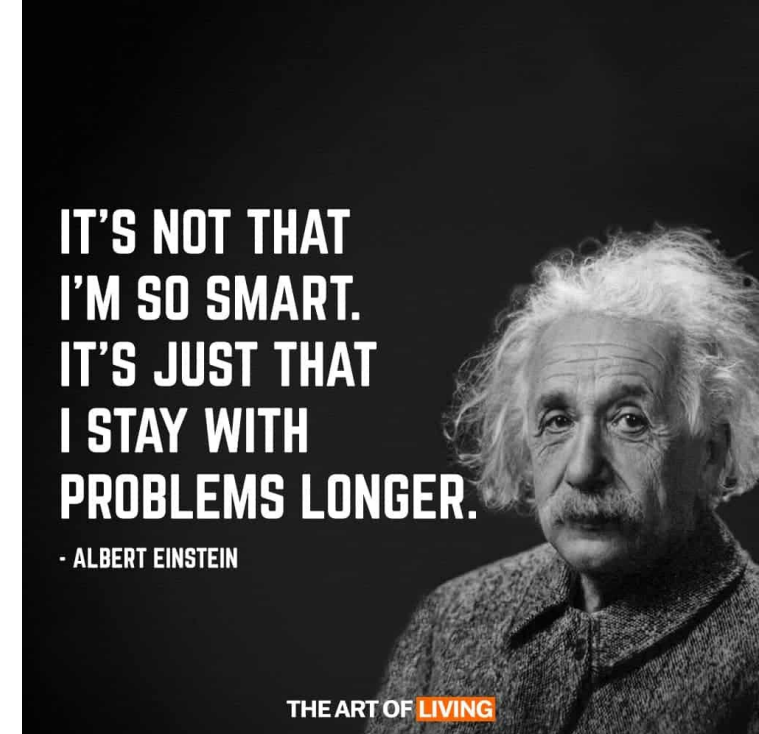
Announcements

TODO Reminders:

Readings are due **before** lecture

- Reading 17 (zybooks) – you should have already done that 😊
- Lab 11
- Reading 18 (zyBooks) – you should have already done that 😊
- Lab 12
- Reading 19 (zybooks) – you should have already done that 😊
- RPA 9

Keep practicing your RPAs in a spaced and mixed manner 😊



<https://theartofliving.com/growth-mindset-quotes/>

Help Desk

Day	Time : Room
Monday	12 PM - 2 PM : CSB 120
Tuesday	6 PM - 8 PM : Teams
Wednesday	3 PM - 5 PM : CSB 120
Thursday	6 PM - 8 PM : Teams
Friday	3 PM - 5 PM : CSB 120
Saturday	12 PM - 4 PM : Teams
Sunday	12 PM - 4 PM : Teams

Recall Activity

- Analyze the class Cake presented and write all concepts and ideas you can remember regarding what is a class and how we can define and use it.
- Make a comment line by line.

```
public class Cake {  
    public static final boolean IS_GOOD = true;  
    private String name;  
    private double cost;  
  
    public void setName(String str) {  
        name = str;  
    }  
    public String getName() {  
        return name;  
    }  
    public void setCost(double cost) {  
        this.cost = cost;  
    }  
    public double getCost() {  
        return this.cost;  
    }  
    public Cake(){  
        this("", 0);  
    }  
    public Cake(String name, double cost) {  
        setName(name);  
        setCost(cost);  
    }  
}
```

Review

- Classes are:
 - Recipes
 - Types (ways to create them)
 - Objects
 - Foundation of Object Oriented Programming
- Classes have:
 - variables
 - methods
 - constructors
- Variables and Methods have:
 - scope
 - Who can access them
 - Memory Type
 - static or instance

```
public class Cake {  
    public static final boolean IS_GOOD = true;  
    private String name;  
    private double cost;  
  
    public void setName(String str) {  
        name = str;  
    }  
    public String getName() {  
        return name;  
    }  
    public void setCost(double cost) {  
        this.cost = cost;  
    }  
    public double getCost() {  
        return this.cost;  
    }  
    public Cake(){  
        this("", 0);  
    }  
    public Cake(String name, double cost) {  
        setName(name);  
        setCost(cost);  
    }  
}
```

Static x Instance Variables

- Static
 - Belongs to the class
 - How do you access a public static variable outside of the class?
 - NameClass.nameStaticVariable
 - Example
 - Cake.IS_GOOD
- Instance
 - Belongs to the object
 - How do you access a private instance variable?
 - You will need to have a get method for each variable that you want to have access from other class
 - nameObject.getNameVariable()
 - Example
 - Cake cake1 = new Cake("chocolate", 3.50);
 - System.out.println(cake1.getName());



```
public class Cake {  
    public static final boolean IS_GOOD = true;  
    private String name;  
    private double cost;  
  
    public void setName(String str) {  
        name = str;  
    }  
    public String getName() {  
        return name;  
    }  
    public void setCost(double cost) {  
        this.cost = cost;  
    }  
    public double getCost() {  
        return this.cost;  
    }  
    public Cake(){  
        this("", 0);  
    }  
    public Cake(String name, double cost) {  
        setName(name);  
        setCost(cost);  
    }  
}
```

Checking your Understanding (Part 1)

- Identify:
 - Class variables (scope and type)
 - Instance variables (scope and type)
- What is the purpose of the class variable in this example?
- How can we access the class variable from another class?
- How can we access the instance variable from another class?

```
public class Store {  
  
    public static int nextId = 101;  
  
    private String name;  
    private String type;  
    private int id;  
  
    public Store(String storeName, String storeType) {  
        name = storeName;  
        type = storeType;  
        id = nextId;  
  
        ++nextId;  
    }  
  
    public int getId(){  
        return id;  
    }  
}
```

Static Methods

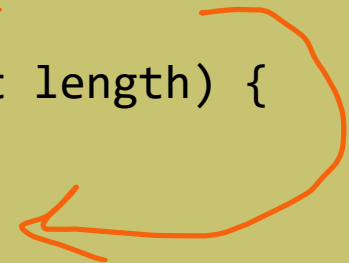
- instance methods
 - Methods that need class level information
 - **Box bx = new Box(10, 10, 10);**
 - **bx.getVolume()**
 - Uses the Box's width, height, length
 - is called on the constructed object
- static method
 - Methods that "self contained"
 - Matches the *concept* of a class, but not unique to object
 - **Box.calcVolume(10, 10, 10);**
 - one-time use
 - Just does 'one thing' and done
 - static may not call instance methods without building an object
 - but instance can call static!

```
public class Box {  
    private int width;  
    private int height;  
    private int length;  
  
    public int getVolume() {  
        return width * height * length;  
    }  
  
    public Box(int width, int height, int length) {  
        this.width = width;  
        this.height = height;  
        this.length = length;  
    }  
  
    public static int calcVolume(int w, int h, int l) {  
        return w * h * l;  
    }  
}
```

Overloaded Constructors

- Just like methods
 - Constructors can be overloaded.
- Standard practice
 - call the most specific constructor with default values
 - `this()` (notice parents) is used to call the constructor.
 - must be **first line** in the constructor.
 - Keep it DRY!
- When you write a constructor with parameters, the default one is not supported anymore!
 - `Box b1 = new Box();` --- Error!
- Really ask yourself
 - What do you need
 - Where do you get it!

```
public class Box {  
    /* ... */  
    public Box(int cubeSize) {  
        // A one parameter constructor that sends default  
        // values to the largest  
        this(cubeSize, cubeSize, cubeSize);  
    }  
    public Box(int width, int height, int length) {  
        this.width = width;  
        this.height = height;  
        this.length = length;  
    }  
    public static void main(String[] args) {  
        Box rec = new Box(10, 20, 10);  
        Box cube = new Box(10);  
    }  
}
```



Checking your Understanding (Part 2)

- Rewrite the class Pet to have its constructors properly overloaded.

```
public class Pet {  
    private String name;  
    private int age;  
  
    public Pet() {  
        name = "Unnamed";  
        age = -1;  
    }  
  
    public Pet(String petName, int yearsOld) {  
        name = petName;  
        age = yearsOld;  
    }  
  
    public String toString() {  
        return name + ", " + age;  
    }  
}
```

Packages

- Is a grouping of related types, classes, interfaces, and subpackage
- Use “import” to add those packages to your program
- `java.lang` is automatically imported in all Java programs
- `import java.io.File;` versus `import java.io.*;`

Package	Sample package members	Description
<i>java.lang</i>	String, Integer, Double, Math	Contains fundamental Java classes. Automatically imported by Java.
<i>java.util</i>	Collection, ArrayList, LinkedList, Scanner	Contains the Java collections framework classes and miscellaneous utility classes.
<i>java.io</i>	File, InputStream, OutputStream	Contains classes for system input and output.
<i>javax.swing</i>	JFrame, JTextField, JButton	Contains classes for building graphical user interfaces.

Unit Testing

- a program whose job is to thoroughly test another program (or portion) via a series of input/output checks known as test cases
- Example: FileTester.java class available in Lab 11!
- <https://github.com/CSU-CompSci-CS163-4/Lab11FileOutput/blob/main/src/FileTester.java>