Lecture 2

Classes and Objects

Some slides were borrowed from Josh Hug and Adam Jundt

Reminders

- Mic
- Make sure to use public wifi

Classes

Motivational Video

- https://www.youtube.com/watch?v=eJrBRjtr0Ro

- Let's design a Car class
    - What do we need to build a car?

Car Class

```
String typeOfCar
String color
int maxSpeed
int damageLevel
--------------------
void accelerate()
void decelerate()
void blowUp()
void takeDamage(int damage)
```

Classes

- **Constructor** – how to set up your object when it’s created
- **State (instance variables)** – what describes your object (variables)
- **Behaviors** – what your object can do (methods)

```
public class Car {
    String typeOfCar;
    String color;
    int maxSpeed;
    int damageLevel;
    public static void accelerate() {
        // does something
    }
    public static void decelerate() {
        // does something
    }
    public static void blowUp() {
        System.out.println(“baaaam!”);
    }
    public static void takeDamage(int damage) {
        damageLevel += damage;
   }
}
```

What is missing?

Can we ran it as-is?

```
public class Car {
     String typeOfCar;
     String color;
     int maxSpeed;
     int damageLevel;
     public static void accelerate() {
          // does something
     }
     public static void decelerate() {
          // does something
     }
     public static void blowUp() {
          System.out.println(“baaaam!”);
     }
     public static void takeDamage(int damage) {
          damageLevel += damage;
    }
}
```

What is missing?

Can we ran it as-is?

Ans: No, the *main* method is missing. Another class will have it.

Simplified Car class

```
public class Car {

   public static void blowUp() {
       System.out.println("baaaam!");
   }
}
```

```
public class CarLauncher {

public static void main(String[] args){
           ???? ;
     }
}
```

How to call a method blowUp() from a Car class?

A: blowUp();

B: Car.blowUp();

C: CarLauncher.blowUp();

D: System.out.println(blowUp);

E: None of the above

Object Instantiation

- Classes can be instantiated as objects.
  - We’ll create a single `Car` class, and then create instances of this class.
  - The class provides a *blueprint* that all `Car` objects will follow.

- By storing different data in *instance* variables.
- Defining different *behaviors* in methods.

Defining a Typical Class (Terminology)

```
public class Car {
 public int damageLevel;

    public Car (int dl){
       damageLevel = dl;
    }

    public void blowUp(){
     .....
    }

}
```

Instance variable. Can have as many of these as you want.

Constructor (similar to a method, but not a method). Determines how to instantiate the class. Has the same name as a class.

Non-static method, a.k.a. Instance Method. Idea: If the method is going to be invoked by an instance of the class (as in the next slide), then it should be non-static.

Roughly speaking: If the method needs to use "**my**" instance variables", the method must be non-static.

How to call a method on an instance of a class?

```
public class Car {
    int damageLevel;

    public void blowUp() { ..}

}
```

A:

```
Car audi;
audi.blowUp();
```

B:

```
Car audi;
audi = new Car();
audi.blowUp();
```

C:

```
Car audi = new Car();
audi.blowUp(10);
```

D:

```
Car audi = new Car();
audi->blowUp();
```

E: B and C

REMINDER

mic

Constructor

- Same name as the class, no return type

```
public Car(String myColor) {
    color = myColor;
}
Car c = new Car("green");
```

- Called automatically by `new` operator
- Often overloaded:
    - Constructor without parameters is called the *default* constructor

Constructor overloaded example

```
public Car(String myColor) {
    color = myColor;
    doors = 4;
}
public Car(String myColor, int doorNum) {
    color = myColor;
    doors  = doorNum;
}
Car c = new Car(“green”, 2);
```

What gets printed

```
public class Chalk {
  String color;
  public Chalk() {
    color = “black”;
  }
  public Chalk(String newColor) {
    color = newColor;
  }
  public void write(String word) {
    System.out.println(“/” + color + “/ “
                          + word);
  }
}
```

```
public class Example{
  public static void main(String[] a) {
    Chalk c1 = new Chalk();
    Chalk c2 = new Chalk(“green”);

    c1.write(“grass”);
    c2.write(“dress”); }
}
```

A) /black/ grass, /black/ dress
B) /black/ grass, /green/ dress
C) /green/ grass, /green/ dress
D) /black/ dress, /black/ grass
E) None of the above