CSC 207
Algorithms and Object-Oriented Design

Weiss, from the beginning of chapter 3 through section 3.2 and sections 3.4 through 3.7 (pages 69–73 and pages 76–90)
Object

• You see real-world objects e.g., dog, desk, and bike.

• Objects have two characteristics: state and behavior
  – For dog, its state is name, color, and breed. Behavior is barking and fetching.

• Real-world objects vary in complexity.
  – Desktop lamp has two states (on, off) but desktop radio has an additional state (on, off, current volume)
Object

- Software object is similar to real-world object.
- An object stores its state in fields and exposes its behavior through methods.
- Methods operate on an object’s internal state.
Benefits of Objects

• **Modularity**: Source code for an object is written and maintained **independent** of other objects.

• **Information hiding**: The **internal implementation** is **hidden** from outside world.

• **Code-reuse**: use an **existing** object in your program.

• **Debugging ease**: Replacing a problematic object with another object.
Class

• In real-world, many objects are of the same kind.

• Many bicycles have the same make and model.

• In object-oriented programming, your bicycle is an instance of a class of objects (bicycles).

• A class is a blueprint from which objects are created.
Class Declaration

• Access level (visibility) modifiers indicate how the outside world use fields or invoke methods.
• **Fields** are usually **private** members. They are not directly accessed by methods of other classes.
• **Methods** are usually **public** members.
• The visibility rule also applies to classes.
Constructor

- Constructor creates (and initializes) objects. It has the same name as the class.
- They are public and do not have return type.
- Constructors can be overloaded.
Test Program

```java
public class TestBicycle {
    public static void main(String[] args) {
        Bicycle myBike = new Bicycle(30, 0, 8);
        myBike.setGear(4);
    }
}
```

• When you run a class file, the main method runs.
• Generally, it’s illegal to define the main method in the same class as the other methods are defined.
this Keyword

• this references the current object (the object whose constructor or method is called).

• Refer to any member of the current object within a constructor or method.

• this can refer to object fields or constructor
Use this with fields

A parameter in a constructor shadows an object field.

```java
public class Point {
    private int x;
    private int y;

    // constructor
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
}
```
Use this with Constructor

- Inside a constructor, call another constructor. Both constructors are in the same class.
- The invocation of another constructor must be the first line in the constructor.

```java
public class Point {
    private int x;
    private int y;

    // constructor without argument
    public Point() {
        this(0, 0);
    }

    // constructor with two arguments
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
}
```
instanceof Operator

• Check whether an object is an instance of a class type.

```java
public class TestPoint {

    public static void main(String[] args) {
        Point obj1 = new Point();

        if(obj1 instanceof Point) {
            System.out.println("obj1 is instanceof Point");
        }
    }
}
```
Instance versus Static Members

• When an object is instantiated from a class, it has its own distinct copies of instance fields.
• An instance field is stored in a different memory location.

• Some fields are common among all object. These fields are called static fields.
• A static field is stored in a fixed location in memory. Any object can change the value of the static field.
• Static fields are usually initialized inside class.
Instance versus Static Members

```java
public class Bicycle {
    // three instance fields
    private int cadence;
    private int gear;
    private int speed;

    // static field
    private static int numberOfBicycles = 0;
}

Inside the main method
Bicycle.numberOfBicycles++;
```
• Static fields are commonly used as constants.
• Constants are spelled with uppercase letters. Words are separated by underscore.

```
public static final int MAX_GEAR = 5;
```
Static Methods

• Static methods are usually invoked by class names.
• A common use of static methods is to access static fields.

    public static int getNumberOfBicycles() {
        return numberOfBicycles;
    }