

CISC 3115 TY3
C06c: "this" and Immutable
Class/Object

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Outline

- Passing objects to methods
- Array of objects
- Scope of variables
- Time permitting
 - The this reference variable
 - Immutable class and object

The this Keyword

- The this keyword is the name of a reference that refers to an object itself.
- Common use
 - To reference a class's hidden data fields.
 - To enable a constructor to invoke another constructor of the same class.

Using this

```
public class F {  
    private int i = 5;  
    private static double k = 0;  
  
    void setI(int i) {  
        this.i = i;  
    }  
  
    static void setK(double k) {  
        F.k = k;  
    }  
}
```

Suppose that f1 and f2 are two objects of F.
F f1 = new F(); F f2 = new F();

Invoking f1.setI(10) is to execute
this.i = 10, where **this** refers f1

Invoking f2.setI(45) is to execute
this.i = 45, where **this** refers f2

Calling Overloaded Constructor

```
public class Circle {  
    private double radius;
```

```
    public Circle(double radius) {  
        this.radius = radius;  
    }
```

→ this must be explicitly used to reference the data field radius of the object being constructed

```
    public Circle() {  
        this(1.0);  
    }
```

→ this is used to invoke another constructor

```
    public double getArea() {  
        return this.radius * this.radius * Math.PI;  
    }  
}
```

↓ ↓
Every instance variable belongs to an instance represented by this, which is normally omitted

Questions?

- What is the "this" reference variable
- What are the two common usage?

Immutable Objects and Classes

- The content of an object cannot be changed once the object is created

Immutable Objects and Classes: Example

- The content of objects of the following Circle class cannot be changed

- Why?

```
public class Circle {  
    private radius = 1.0;  
    public Circle() {  
    }  
    private double getArea() {  
        return radius * radius * Math.PI;  
    }  
}
```


Mutators

- Mutators: methods that changes the value of data fields
- A class with all private data fields and without mutators is not necessarily immutable.
 - A data field can be a reference variable whose content can be changed with the reference

No Mutator, but Immutable: Example

```
import java.util.Date;

public class Student {
    private int id;
    private String name;
    private Date dateCreated;

    public Student(int ssn, STring newName) {
        id = ssn;
        name = newName;
        dateCreated = new Date();
    }

    public int getId() {
        return id;
    }
}
```

```
public String getName() {
    return name;
}

public Date getDateCreated() {
    return dateCreated;
}
}

...

public static void main(String[] args) {
    Student s = new Student(123, "John");
    Date d = s.getDateCreated();
    d.setTime(200000);
}
```

What Class is Immutable?

- These conditions must hold
 - mark all data fields private
 - provide no mutator methods
 - no accessor methods that would return a reference to a mutable data field object.

Questions?

- Concept of immutable classes and objects
- Concept of mutators
- Condition under which a class or a object is immutable