

Lecture 7

Strings & Stepwise Refinement

Announcements for This Lecture

Readings

- pp. 175–181
- Sections 2.5, 3.1.2-3.1.3
- (optional) PLive p. 2-5

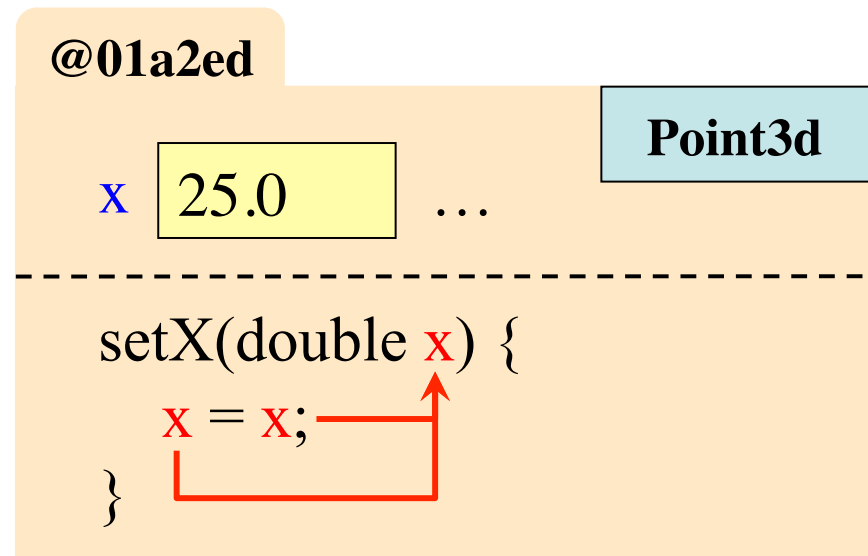
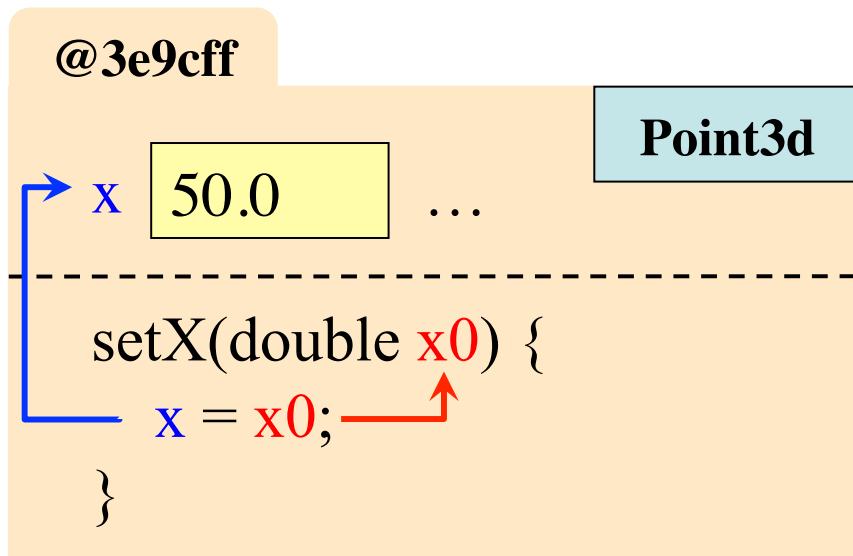


Assignments

- Assignment 1 due tonight
 - Before Midnight!!!
 - Will have get by class Thu
 - Revise if you are told
- New Assignment Posted
 - No code; written only
 - Meant to do while you revise
 - Due in class next week
- Will go to 2-week assignment schedule after Assignment 3

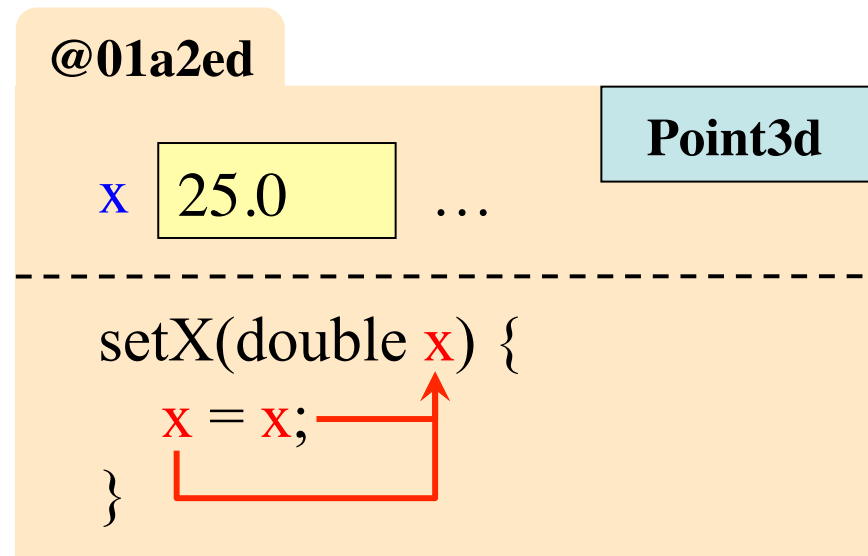
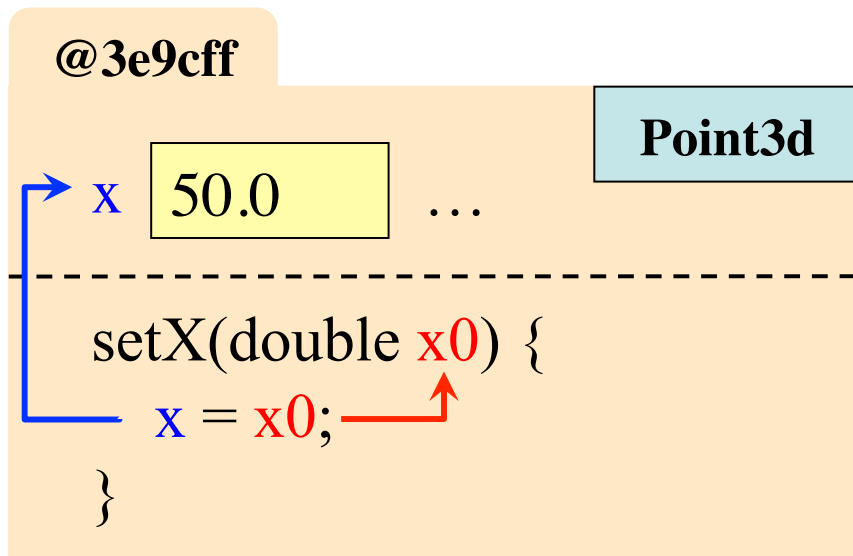
Inside-Out Rule (See p. 83)

- Methods reference fields or static variables (of same class) **3**
 - Can reference parameters of that method **2**
 - Can reference local variables inside same braces `{}` **1**
- If two of the same name, use the **closest** declaration



Inside-Out Rule (See p. 83)

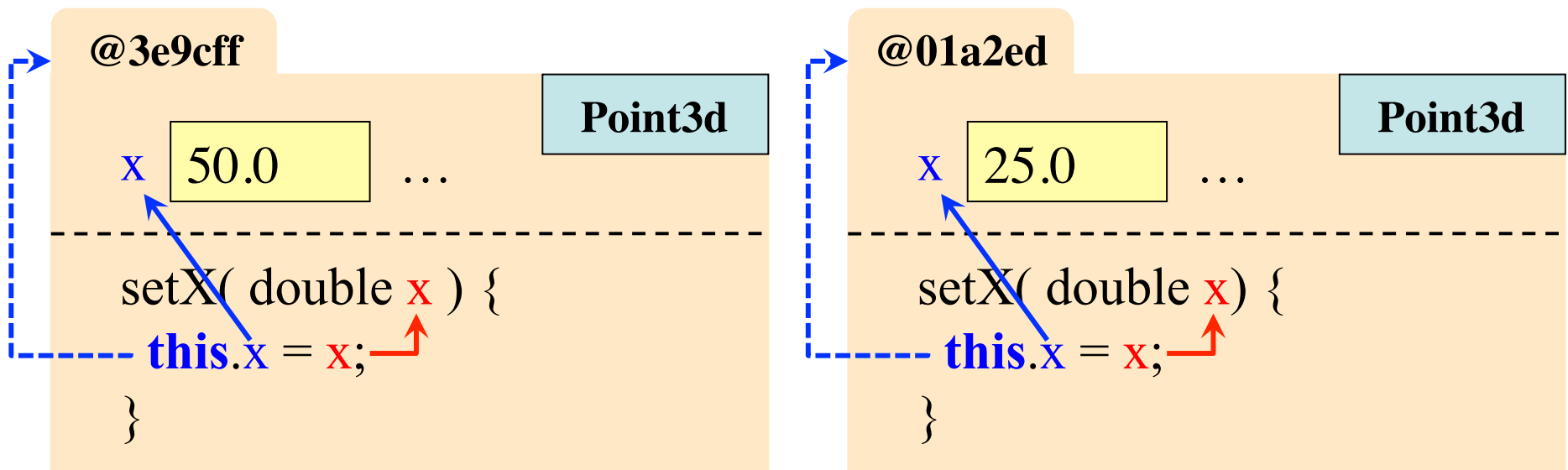
- Parameter **x0** is found in the frame for the method call. Exists temporarily
- Parameter **x** “blocks” (or **shadows**) the reference to the field **x**.



A Solution: `this`

`this` is a built-in “variable” that gives an object name

- In object (folder) `@3e9cff`, `this` refers to `@3e9cff`
- In object (folder) `@01a2ed`, `this` refers to `@01a2ed`



How Well Are You Following?

```
public class Demo {  
    private int var;  
  
    public Demo() {  
        var = 0;  
    }  
  
    public int getSetWTF(boolean flag) {  
        if (flag) {  
            int var;  
            var = 1;  
        } else {  
            var = 2;  
        }  
        return var;  
    }  
}
```

- Function (not procedure) that “does something”
- This is called a function with “**side-effects**”

- Create a new Demo object:
Demo d = new Demo();

- What does this give us?

d.getSetWTF(true)

- A: Function gives 0
- B: Function gives 1
- C: Function gives 2
- D: I have no clue

String is a Class; Quoted Text is an Object

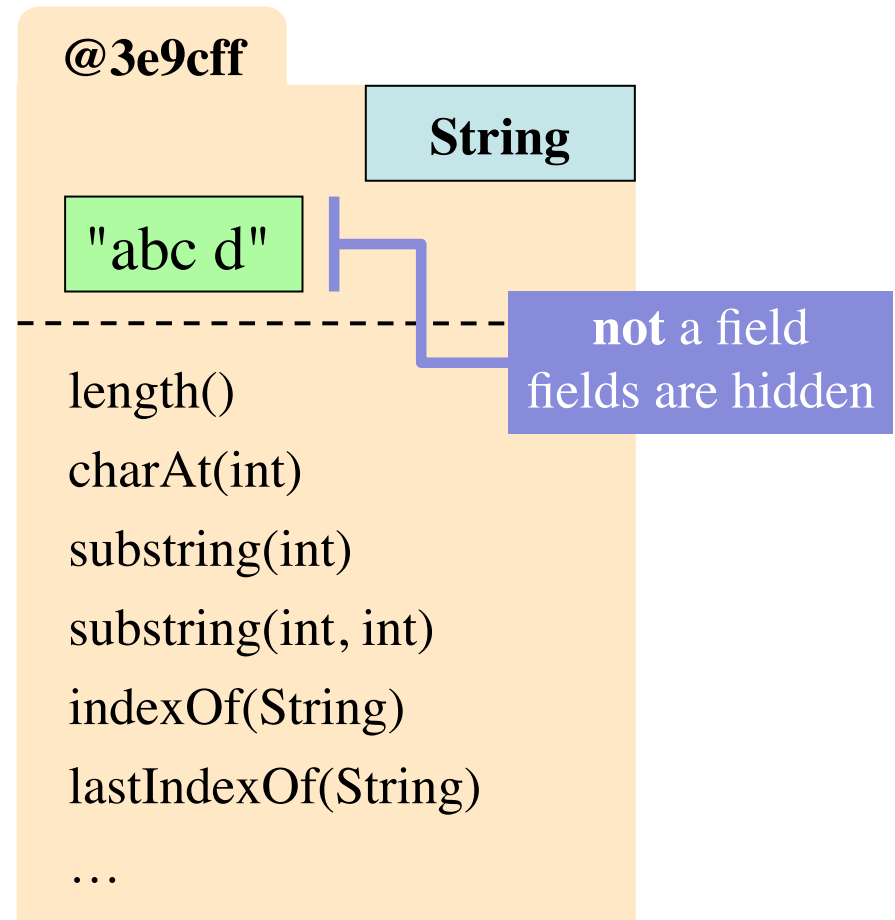
- `String s = "abc d";`
- Indexed characters:

0 1 2 3 4

abc d

- `s.length()` is 5
- `s.charAt(2)` is 'c'
- `s.substring(2)` is "c d"
- `s.substring(1,3)` is "bc"

s @3e9cff



String Has a Lot of Useful Methods

- `String s = "abc d";`
- Indexed characters:

0 1 2 3 4

abc d

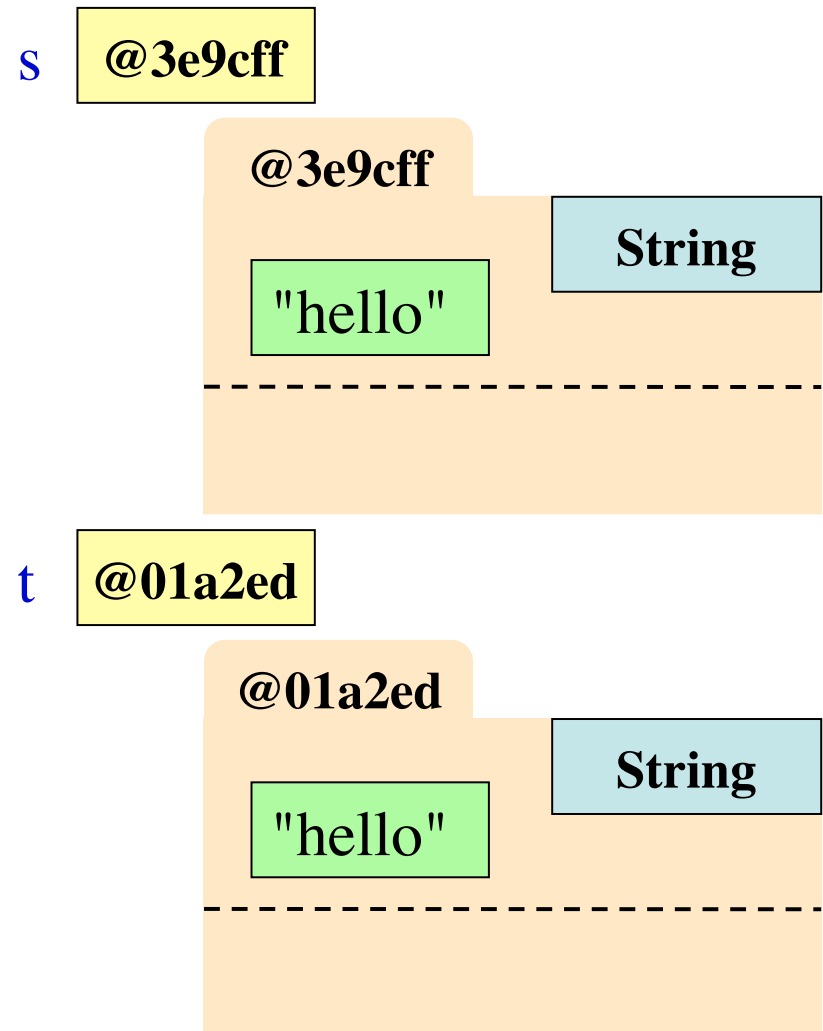
- See text pp. 175–181
- Look in CD ProgramLive
- Look at API specs for String

- `s.substring(2,4)` is `"c "` (**NOT** `"c d"`)
- `s.substring(2)` is `"c d"`
- `" bcd ".trim()` is `"bcd"`
(trim beginning and ending blanks)
- `s.indexOf("bc")` is 1
(index or position of first occurrence of in `"bc"` or -1 if none)

String Variables Hold Folder Names

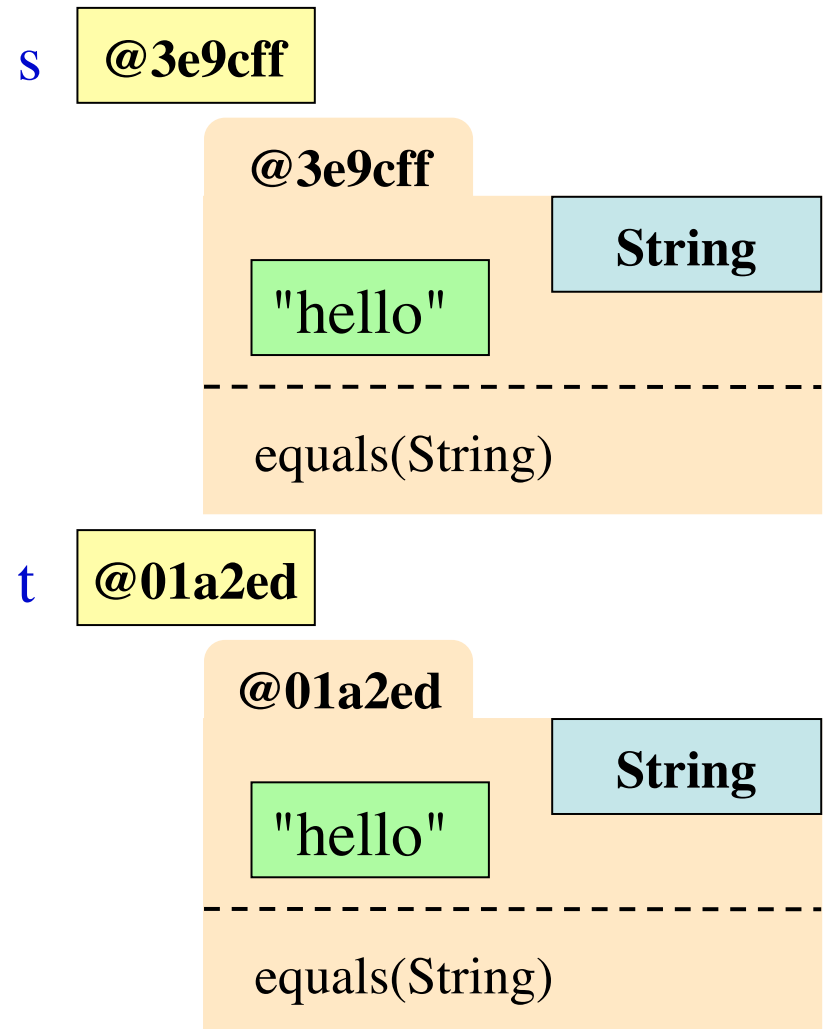
- Create two Strings
 - `String s = "hello";`
 - `String t = "hello";`
- What is the value `s == t`?

A: true
B: false
C: neither
D: I do not know



String Variables Hold Folder Names

- Create two Strings
 - `String s = "hello";`
 - `String t = "hello";`
- Do not use `==` to test equality of `s` and `t`
 - `s == t` tests if same object
 - Not useful for Strings
- Use `equals()` instead
 - `s.equals(t)` tests if they have the same text



Algorithms: Heart of Computer Science

- **Algorithm:** A step-by-step procedure for how to do something (usually a calculation).
- **Implementation:** How to write an algorithm in a specific programming language
- Good programmers know how to separate the two
 - Work out algorithm on paper or in head
 - Once done, implement it in the language
 - Limits errors to **syntax errors** (easy to find), not **conceptual errors** (much, much harder to find)
- Key to designing algorithms: **stepwise refinement**

Algorithms: Heart of Computer Science

- **Algorithm:** A step-by-step procedure for how to do something (usually a calculation).
- **Implementation:** How to write an algorithm in a specific programming language

Java does what you say,
not what you meant

Java cannot
“understand” you

- **Two**
 - Once done, implement it in the language
 - Limits errors to **syntax errors** (easy to find), not **conceptual errors** (much, much harder to find)
- Key to designing algorithms: **stepwise refinement**

Stepwise Refinement: Basic Principles

- **Write Specifications First**
Write a method specification before writing its body
- **Take Small Steps**
Do a little at a time; follow the **Mañana Principle**
- **Compile Often**
This can catch syntax errors
- **Separate Concerns**
Focus on one step at a time
- **Intersperse Programming and Testing**
When you finish a step, test it immediately

Mañana Principle

- If not in current step, delay to “tomorrow”
 - Use **comments** to write steps in English
 - Add “**stubs**” to ensure the program compiles (e.g. empty definitions or bogus return statements)
 - Slowly replace stubs/comments with real code
- Only create new local variables if you have to
- Sometimes results in creation of more methods
 - Replace the step with a method call
 - But leave the *method definition* empty for now
 - This is called **top-down design**

Example: Reordering a String

- `lastNameFirst("Walker White")` is "White, Walker"

```
/** Yields: copy of s but in the form <last-name>, <first-name>  
 * Precondition: s is in the form <first-name> <last-name>  
 *               with one blank between the two names */
```

```
public static String lastNameFirst(String s) {  
    // Find the first name  
    // Find the last name  
    // Put them together with a comma  
    return ""; // Stub return  
}
```

Example: Reordering a String

- `lastNameFirst("Walker White")` is "White, Walker"

```
/** Yields: copy of s but in the form <last-name>, <first-name>  
 * Precondition: s is in the form <first-name> <last-name>  
 *               with one blank between the two names */
```

```
public static String lastNameFirst(String s) {  
    int endOfFirst = s.indexOf(" ");  
    String firstName = s.substring(0,endOfFirst);  
    // Find the last name  
    // Put them together with a comma  
    return firstName; // Stub return (which you can test!)  
}
```


Refinement: Creating Helper Methods

```
/**
 * Yields: copy of s but in the form
 * <last-name>, <first-name>
 * Precondition: s is in the form
 * <first-name> <last-name>
 * with one blank between names
 */
public static String
lastNameFirst(String s) {
    String first = firstName(s);
    // Find the last name
    // Put together with comma
    return first; // Stub
}
```

```
/**
 * Yields: first name in s
 * Precondition: s is in the form
 * <first-name> <last-name>
 * with one blank between names
 */
public static String
firstName(String s) {
    int end = s.indexOf(" ")
    return s.substring(0,end);
}
```

Refinement: Creating Helper Methods

Do This Sparingly

- If you might use this step in **another** method later
- If implementation is rather long and complicated

```
public static String
lastNameFirst(String s) {
    String firstName = firstName(s);
    // Find the last name
    // Put together with comma
    return firstName; // Stub
}
```

```
/**
 * Yields: first name in s
 * Precondition: s is in the form
 * <first-name> <last-name>
 * with one blank between names
 */
public static String
firstName(String s) {
    int end = s.indexOf(" ")
    return s.substring(0,end);
}
```

Example: Reordering a String

- `lastNameFirst("Walker White")` is "White, Walker"

```
/** Yields: copy of s but in the form <last-name>, <first-name>  
 * Precondition: s is in the form <first-name> <last-name>  
 * with one or more blanks between the names */
```

```
public static String lastNameFirst(String s) {  
    // Find the first name  
    // Find the last name  
    // Put them together with a comma  
    return ""; // Stub return  
}
```