

# Lecture 15:

## Review of Aliasing & Mutability, Floating Point Calculations

---

Last time:

1. Unit testing and JUnit
2. Constructors revisited
3. equals

Today:

1. Project #4 assigned
2. Aliasing and Mutability
3. Floating Point calculations
4. Example class development: Rational Numbers





# Project #4 Is Assigned

- It is due **Wednesday, 10/31 at 11:00PM**
- The project is **closed**
  - You must complete the project by yourself
  - Assistance can only be provided by teaching assistants (TAs) and instructors
  - You must not look at other students' code
- **Start now!**
  - Read entire assignment from beginning to end before starting to code
  - Check out assignment now from CVS
  - Follow the instructions *exactly*



# Taking Care of Corner Cases

- FancyWord example from Friday
  - String of "" was a corner case that we needed to test for
  - Write new test cases or new asserts in the test cases that already exist to take care of this
- What about null references as corner cases?

```
public void testNullAndEmpty() {  
    FancyWord a = new FancyWord(null);  
    assertEquals(null, a.toString());  
    FancyWord b = new FancyWord("");  
    assertEquals("", b.toString());  
}
```

# What about Strings and Aliasing?

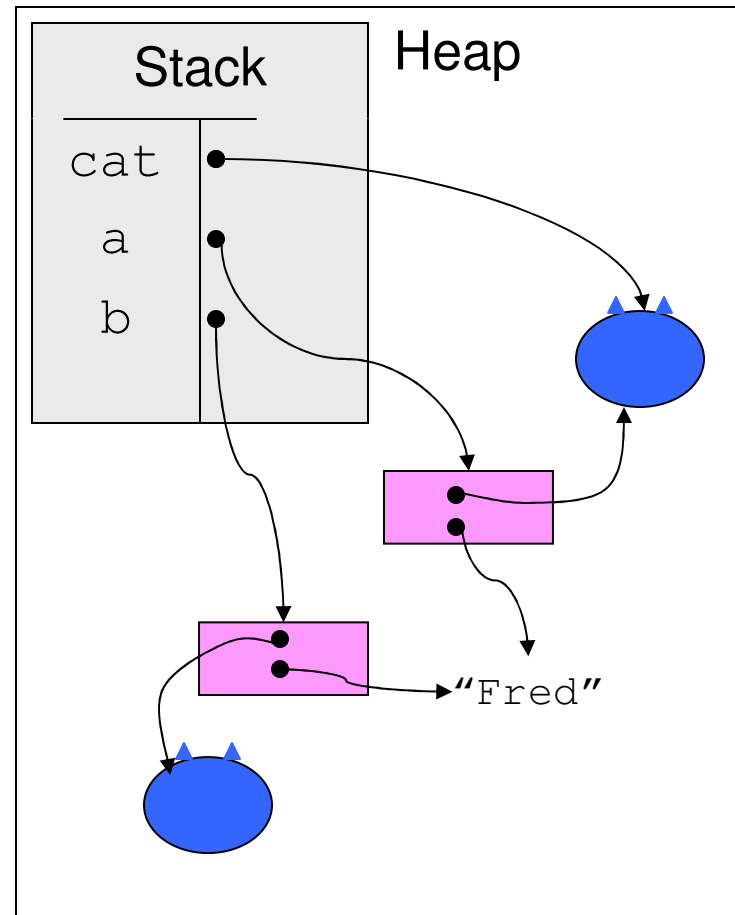
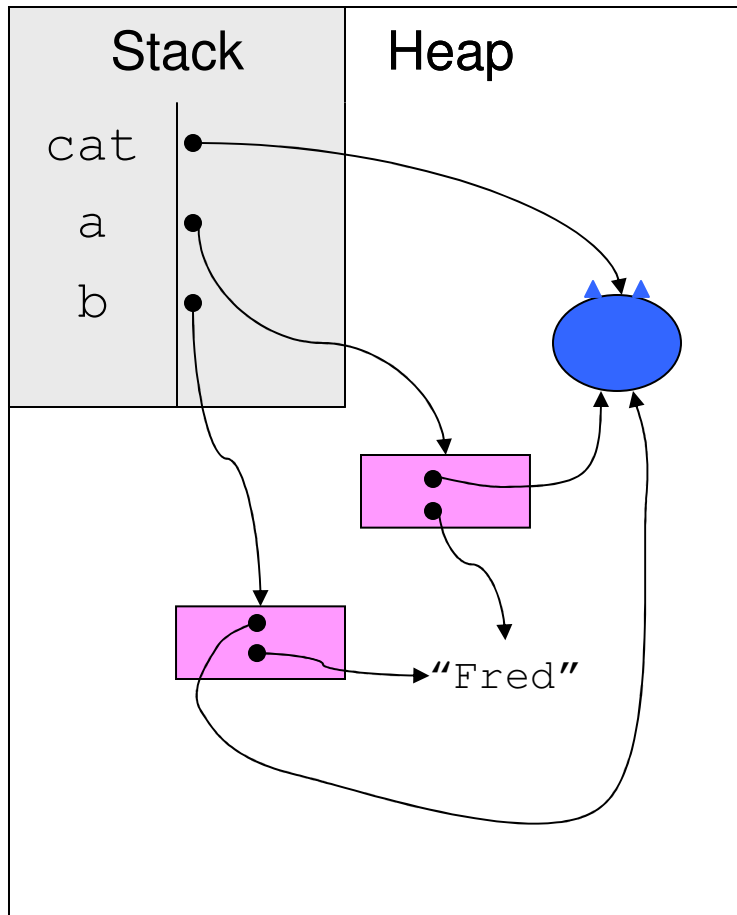


- `String` objects are *immutable*; fields cannot be changed once created
  - **Mutable** objects: fields (values of instance variables) can be changed (e.g. `Cat`, `Student`, etc.)
  - **Immutable** objects: fields (values of instance variables) cannot be changed
  - See `String` API: <http://java.sun.com/j2se/1.3/docs/api/java/lang/package-summary.html>
- In this example:
  - `y` is created as an alias for `String x`
  - `b` is created as an alias for `Cat a`
  - but the results are different.

```
x = Jan Plane  
y = Jan
```

```
=====  
Fluffy has been created!  
Fluffy has been eaten!  
Fluffy has 8 more lives...  
a = Fluffy ( 8 lives.)  
b = Fluffy ( 8 lives.)
```

# Which picture represents the current status of memory?





# Floating Point Calculations

What will this print?

```
public class SimpleMath {
    public static void main(String[] args) {
        if (3.9 - 3.8 == 0.1) {
            System.out.println("I am a very smart computer.");
        } else {
            System.out.println("I can't do simple arithmetic.");
        }
    }
}
```

- I can't do simple arithmetic.
- Why?
  - Conversion of floating point to binary leads to precision errors!
  - What can we do?

# Floating Point Calculations (cont.)



Two important rules:

- You can never use `==` to compare floating point values. Instead, check if two numbers are within a certain tolerance of each other.
- Never use floating point values to represent money, e.g., 3.52 to represent \$3.52. Instead, use integer 352 to represent 352 pennies.