

# Object Oriented Programming

## Week 8 Part 3 Refactoring Console Input and Output

# Lecture

- Refactoring Design
- Problem with I/O Design
- Redesign of I/O
- Refactoring Code

# Refactoring Design

# Refactoring

- Refactoring means altering code without altering behavior
- Unit tests help insure that we have not altered behavior
  - If the unit tests continue to pass, we are more confident that the behavior has been preserved

# Changing Design

- Sometimes, in refactoring, we discover flaws in our designs
- In these cases, we may need to alter the unit tests
  - Classes are usually the units
  - Changing design changes the classes
- Still, the unit tests provide a template for correct behavior
  - We try to alter them as little as possible

# Animal I/O Design Flaw

# Rationale for Existing Design

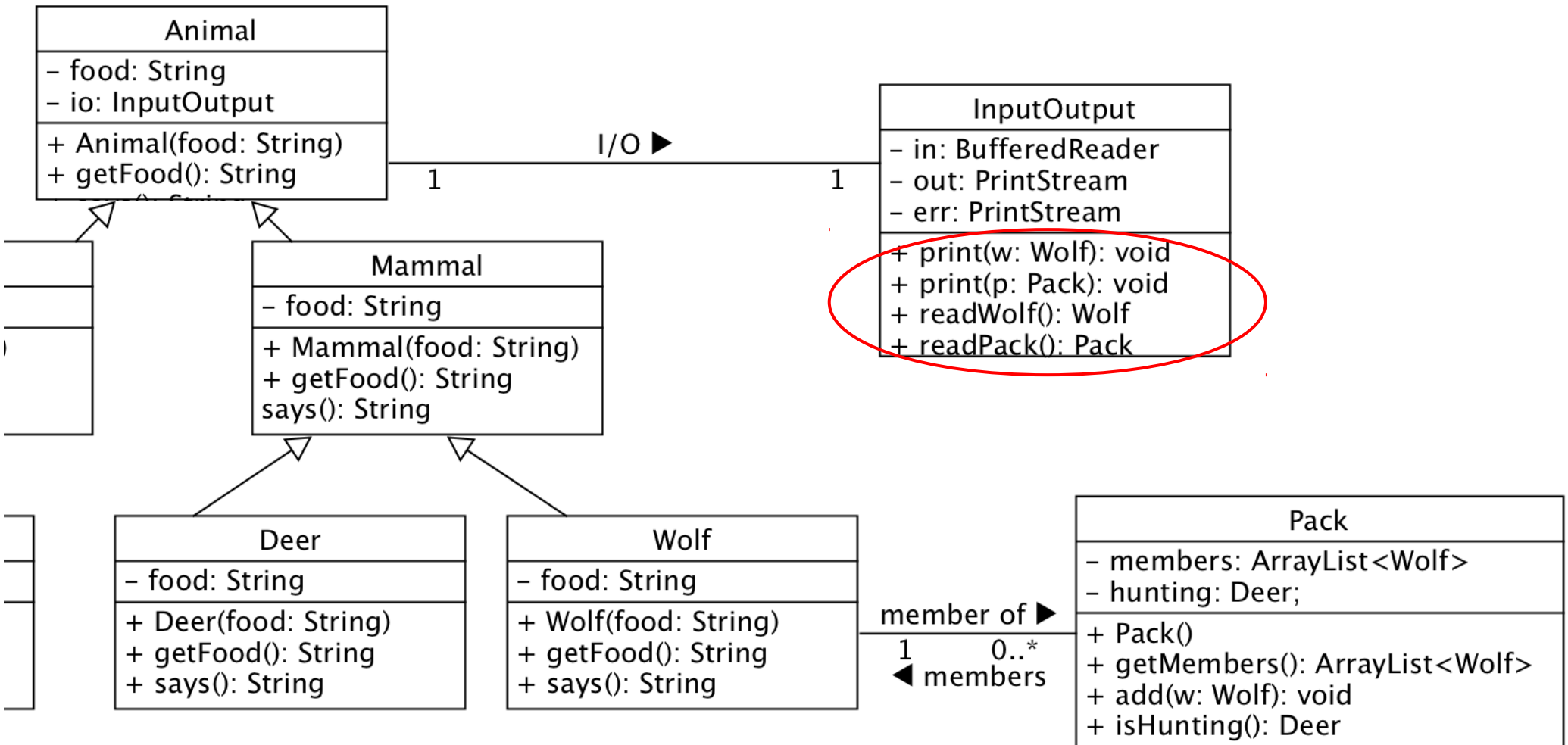
- Demonstrating I/O is easier if everything is in the same class
  - Need not alter more than one class
  - Need not develop more than one unit test

# Problem with Existing Design

- The existing design includes an I/O class that holds all of the print functions
- This class is highly coupled to all of the classes that need to be printed
  - It needs to know all of the fields that need to be printed and how they should be printed
- We want to reduce coupling



# Existing Design

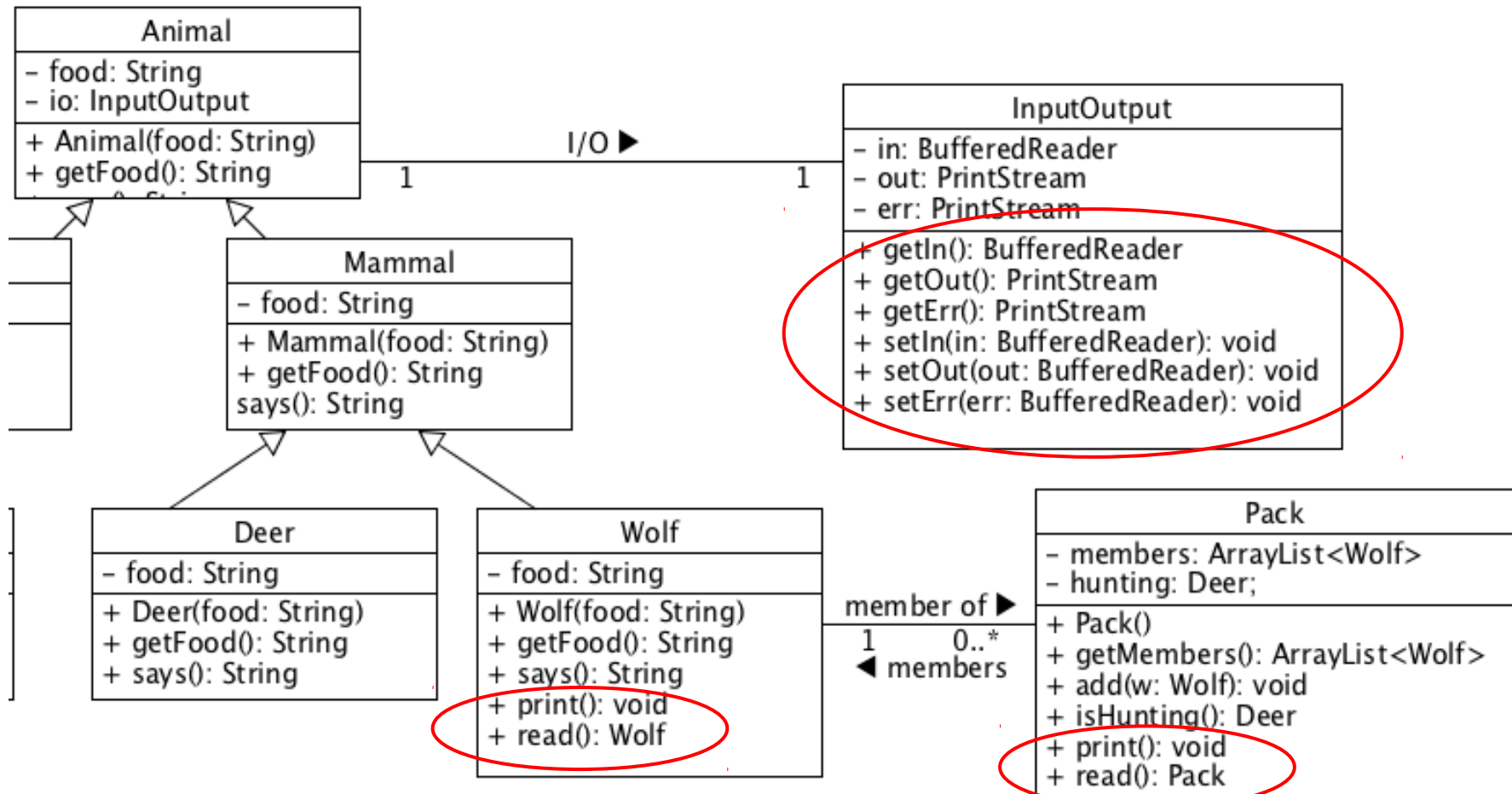


Highly Coupled Design

# Improved Design

- We can improve the design by having each of the objects that need to be printed print themselves
  - Decisions about display should be made by the object themselves
    - Knowledge of the object should be encapsulated in the object
  - Adding printing ability requires only altering the object to be printed
- The AnimalInOut class provides getters and setters for the streams

# Improved Design UML



# Indications of Improved Design

- `InputOutput` is now simply a Java Bean
  - It has fields, getters and setters only
- We do not need to specify what type of object we are reading
  - The type is defined by the object on which it is called
- We can add printing and reading to any new object without changing the `InputOutput` class

# Refactoring Code

# Refactoring tasks

- Alter TestAnimalsInOut to test the altered AnimalsInOut
- Change TestAnimalsInOut to satisfy the tests
- Modify TestWolf check print
  - Can use tests from TestAnimalsInOut
- Modify Wolf to print itself
- Modify TestWolf to check read
- Modify Wolf to read itself

# Refactoring tasks

- Alter TestAnimalsInOut to test the latered AnimalsInOut
- Change TestAnimalsInOut to satisfy the tests
- Modify TestWolf check print
  - Can use tests from TestAnimalsInOut
- Modify Wolf to print itself
- Modify TestWolf to check read
- Modify Wolf to read itself

# Refactoring tasks (cont.)

- Modify TestPack to check print
- Modify Pack to print itself
- Modify TestPack to check read
- Modify Pack to read itself



# Step 1: Removing Print and Read Tests from AnimalInOutTest

- Move testPrintWolf() to TestWolf
- Move testReadWolf() to TestWolf
- Comment out testPrintWolf and testReadWolf
- Move testPrintPack() to TestPack
- Move testReadPack() to TestPack
- Comment out testPrintPack and testReadPack

# New TestAnimalsInOut

```
public class TestAnimalsInOut {  
  
    AnimalsInOut aio = null;  
    ByteArrayOutputStream outString = null;  
    ByteArrayOutputStream errString = null;  
    BufferedReader in = null;  
    PrintStream out = null;  
    PrintStream err = null;  
  
    @Before  
    public void setUp() throws Exception {  
        in = new BufferedReader(new StringReader("test"));  
        outString = new ByteArrayOutputStream();  
        out = new PrintStream(outString);  
        errString = new ByteArrayOutputStream();  
        err = new PrintStream(errString);  
  
        aio = new AnimalsInOut(in, out, err);  
    }  
  
    @Test  
    public void testConstructor() {  
        String s = aio.readLine();  
        assertEquals("test", s);  
        System.out.println("Read from AnimalsInOut: " + s);  
  
        aio.print("test");  
        assertEquals("test", outString.toString());  
        System.out.println("Printed to AnimalsInOut: "  
            + outString.toString());  
  
        aio.printErr("test");  
        assertEquals("test", errString.toString());  
        System.out.println("Printed to AnimalsInOut: "  
            + errString.toString());  
    }  
}
```

Print and Read tests remove

# Add to TestWolf

```
// @Test
// public void testPrintWolf() {
//     Wolf w = new Wolf("Meat");
//
//     aio.print(w);
//     assertEquals("Wolf howls, eats Meat", outString.toString());
//     System.out.println("printWolf wrote: " + outString.toString());
// }
//
// @Test
// public void testReadWolf() {
//     BufferedReader in = new BufferedReader(new StringReader("Deer"));
//     outString = new ByteArrayOutputStream();
//
//     aio = new AnimalsInOut(in, out, err);
//     Wolf w = null;
//
//     w = aio.readWolf();
//     assertEquals("Wolf howls, eats Deer", w.toString());
//     System.out.println("readWolf returned: " + w);
// }
//
```

# Add to TestPack

```
// @Test
// public void testPrintPack() {
//     Pack p = new Pack();
//
//     p.addWolf(new Wolf("Meat"));
//     p.addWolf(new Wolf("Meat"));
//     p.addWolf(new Wolf("Meat"));
//     aio.print(p);
//     assertEquals("Pack contains 3 wolves", outString.toString());
//     System.out.println("testPrintPack() returned: " + outString.toString());
// }
//
// @Test
// public void testReadPack() {
//     BufferedReader in = new BufferedReader(new StringReader("Deer\nDeer\nMeat"));
//     outString = new ByteArrayOutputStream();
//
//     aio = new AnimalsInOut(in, out, err);
//     Pack p = null;
//
//     p = aio.readPack();
//     assertEquals("Pack contains 3 wolves", p.toString());
//     System.out.println("readPack returned: " + p);
// }
// }
```

# Step 2: Add getters and setters to TestAnimalInOut

- Add Test of getter or setter
- Write getter or setter
- Until getters and setters for in, out and err

# Add Test to AnimalsInOutTest

```
@Test
public void testGetIn() {
    BufferedReader br = aio.getIn();
    try {
        String s = br.readLine();
        assertEquals("test", s);
        System.out.println(s);
    } catch (IOException e) {
        fail("BufferedReader exception: " + e.getMessage());
        e.printStackTrace();
    }
}
```

Create method 'getIn()' in type 'AnimalsInOut'

Add cast to 'aio'

Rename in file (%2 R)

...  
return p;  
}

```
public BufferedReader getIn() {
    // TODO Auto-generated method stub
    return null;
}
...
```

Press 'Tab' from proposal table or click for focus

# Add getIn to AnimalsInOut

```
public BufferedReader getIn() {  
    return in;  
}
```

# Test getIn()

▼ animals.TestAnimalsInOut [Runner: JUnit 4] (0.001 s)  
 ✓ testGetIn (0.001 s)  
 ✓ testConstructor (0.000 s)

```
@Test
public void testGetIn() {
    BufferedReader br = aio.getIn();
    try {
        String s = br.readLine();
        assertEquals("test", s);
        System.out.println("getIn().readLine(): " + s);
    } catch (IOException e) {
        fail("BufferedReader from getIn() failed");
        e.printStackTrace();
    }
}
```

Output

Console Problems Java  
<terminated> TestAnimalsInOut [JUnit] /  
getIn().readLine(): test  
Read from AnimalsInOut: test  
Printed to AnimalsInOut: test  
Printed to AnimalsInOut: test  
|



# Add testGetOut() and testGetErr()

```
@Test
public void testGetOut() {
    aio.getOut().print("test");
    // ...
}
```

as se Change to 'getIn(..)'  
Syst Create method 'getOut()' in type 'AnimalsInOut'  
Add cast to 'aio'  
Rename in file (⌘2 R)

```
...
public void testGetOut() {
    aio.getIn().print("test");
    assertEquals("test", outString.toString());
    ...
}
```

Press 'Tab' from proposal table or click for focus

```
@Test
public void testGetErr() {
    aio.getErr().print("test");
    // ...
}
```

as se Change to 'getIn(..)'  
Syst Create method 'getErr()' in type 'AnimalsInOut'  
Add cast to 'aio'  
Rename in file (⌘2 R)

```
...
public void testGetErr() {
    aio.getIn().print("test");
    assertEquals("test", outString.toString());
    ...
}
```

Press 'Tab' from proposal table or click for focus

# Add Methods

```
public PrintStream getOut() {  
    return out;  
}  
  
public PrintStream getErr() {  
    return err;  
}
```

# Run Tests

▼ animals.TestAnimalsInOut [Runner: JUnit 4] (0.000 s)

- testGetIn (0.000 s)
- testGetErr (0.000 s)
- testGetOut (0.000 s)
- testConstructor (0.000 s)

Output

```
@Test
public void testGetOut() {
    aio.getOut().print("test");
    assertEquals("test", outString.toString());
    System.out.println("getOut.print: "
        + outString.toString());
}
```

Output

```
@Test
public void testGetErr() {
    aio.getErr().print("test");
    assertEquals("test", errString.toString());
    System.out.println("getErr.print: "
        + errString.toString());
}
```

Console Problems Java

```
<terminated> TestAnimalsInOut [JUnit]
getIn().readLine(): test
getErr.print: test
getOut.print: test
Read from AnimalsInOut: test
Printed to AnimalsInOut: test
Printed to AnimalsInOut: test
```

# New AnimalsInOut

```
public class AnimalsInOut {  
  
    private BufferedReader in = null;  
    private PrintStream out = null;  
    private PrintStream err = null;  
  
    public AnimalsInOut(BufferedReader in, PrintStream out, PrintStream err) {  
        this.in = in;  
        this.out = out;  
        this.err = err;  
    }  
  
    public String readLine() {  
  
        String s = "uninitialized";  
  
        try {  
            s = in.readLine();  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
  
        return s;  
    }  
  
    public void print(String string) {  
        out.print(string);  
    }  
  
    public void printErr(String string) {  
        err.print(string);  
    }  
  
    public BufferedReader getIn() {  
        return in;  
    }  
  
    public PrintStream getOut() {  
        return out;  
    }  
  
    public PrintStream getErr() {  
        return err;  
    }  
}
```

# Updating Wolf

# Tasks to update Wolf

- Add testPrint to TestWolf
- Add print() to Wolf
- Run testPrint()
- Add testRead to TestWolf
- Add read() to Wolf
- Run testRead()

# Add testPrint to TestWolf

Create Streams

Create AnimalsInOut

Print Wolf on AnimalsInOut

```
@Test
public void testPrintWolf() {
    BufferedReader in = new BufferedReader(new StringReader("test"));
    ByteArrayOutputStream outString = new ByteArrayOutputStream();
    PrintStream out = new PrintStream(outString);
    ByteArrayOutputStream errString = new ByteArrayOutputStream();
    PrintStream err = new PrintStream(errString);

    AnimalsInOut aio = new AnimalsInOut(in, out, err);
    Wolf w = new Wolf("Meat");

    w.print(aio);
}

as Create method 'print(AnimalsInOut)' in type 'Wo...
Sy () Add cast to 'w'
    Rename in file (§2 R)

return says() + ", eats " + super.getFood();
}

public void print(AnimalsInOut aio) {
    // TODO Auto-generated method stub
}

}

Press 'Tab' from proposal table or click for focus
```

# Add print to Wolf

```
public void print(AnimalsInOut aio) {  
    aio.print(this.says() + ", eats " + this.getFood());  
}
```



# Test print()

▼ animals.TestWolf [Runner: JUnit 4] (0.001 s)  
 ✓ testHunts (0.000 s)  
 ✓ testSays (0.001 s)  
 ✓ testPrintWolf (0.000 s)  
 ✓ testMemberOf (0.000 s)  
 ✓ testSetMemberOf (0.000 s)  
 ✓ testConstructor (0.000 s)

```
@Test
public void testPrintWolf() {
    BufferedReader in = new BufferedReader(new StringReader("test"));
    ByteArrayOutputStream outString = new ByteArrayOutputStream();
    PrintStream out = new PrintStream(outString);
    ByteArrayOutputStream errString = new ByteArrayOutputStream();
    PrintStream err = new PrintStream(errString);

    AnimalsInOut aio = new AnimalsInOut(in, out, err);
    Wolf w = new Wolf("Meat");

    w.print(aio);
    assertEquals("Wolf howls, eats Meat", outString.toString());
    System.out.println("printWolf wrote: " + outString.toString());
}
```

Output

Console Problems Javadoc D

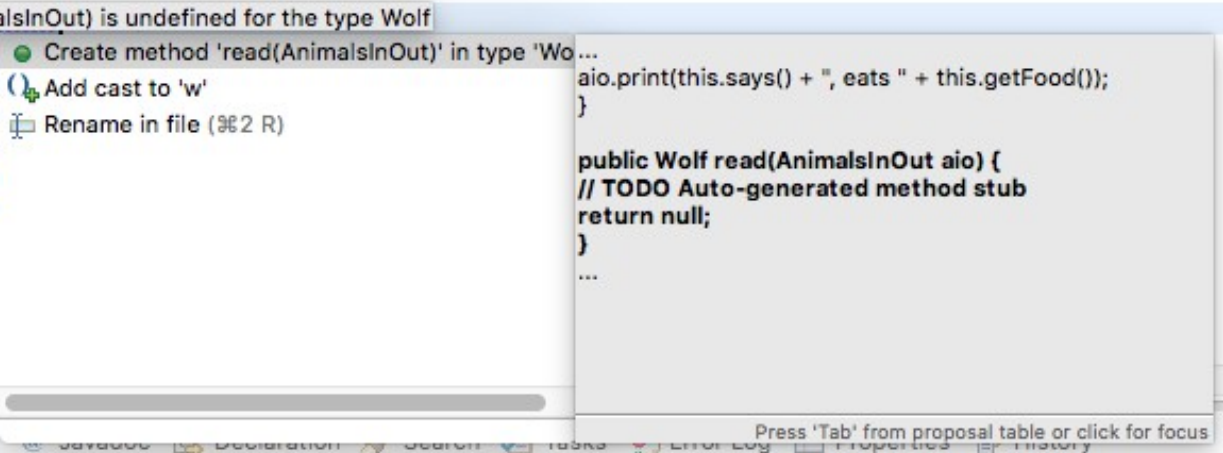
```
<terminated> TestWolf [JUnit] /Library/Java/JavaVir
printWolf wrote: Wolf howls, eats Meat
```

# Add testRead to TestWolf

```
@Test
public void testReadWolf() {
    BufferedReader in = new BufferedReader(new StringReader("test"));
    ByteArrayOutputStream outString = new ByteArrayOutputStream();
    PrintStream out = new PrintStream(outString);
    ByteArrayOutputStream errString = new ByteArrayOutputStream();
    PrintStream err = new PrintStream(errString);

    AnimalsInOut aio = new AnimalsInOut(in, out, err);
    Wolf w = new Wolf("Meat");

    try {
        method read(AnimalsInOut) is undefined for the type Wolf
    } catch (IOException e) {
        fail("IOException: " + e.getMessage());
    }
    assertEquals("Wolf read", "Meat", outString.toString());
    System.out.println("Test passed");
}
```



The screenshot shows an IDE window with a code completion popup. The popup lists several suggestions for the 'read' method in the 'Wolf' class. The first suggestion is 'Create method 'read(AnimalsInOut)' in type 'Wolf...'. The second suggestion is 'Add cast to 'w'', which is highlighted. The third suggestion is 'Rename in file (%2 R)'. The fourth suggestion is 'aio.print(this.says() + ", eats " + this.getFood());'. The fifth suggestion is 'public Wolf read(AnimalsInOut aio) { // TODO Auto-generated method stub return null; } ...'. The IDE interface includes a 'Problems' tab and a status bar at the bottom with the text 'Press 'Tab' from proposal table or click for focus'.

# Add read(AnimalsInOut aio) to Wolf

```
public Wolf read(AnimalsInOut aio) throws IOException {  
    this.setFood(aio.getIn().readLine());  
    return this;  
}
```

# Run Test

▼ animals.TestWolf [Runner: JUnit 4] (0.000 s)  
 ✓ testHunts (0.000 s)  
 ✓ testSays (0.000 s)  
 ✓ testPrintWolf (0.000 s)  
 ✓ testReadWolf (0.000 s)  
 ✓ testMemberOf (0.000 s)  
 ✓ testSetMemberOf (0.000 s)  
 ✓ testConstructor (0.000 s)

```
@Test
public void testReadWolf() {
    BufferedReader in = new BufferedReader(new StringReader("test"));
    ByteArrayOutputStream outString = new ByteArrayOutputStream();
    PrintStream out = new PrintStream(outString);
    ByteArrayOutputStream errString = new ByteArrayOutputStream();
    PrintStream err = new PrintStream(errString);

    AnimalsInOut aio = new AnimalsInOut(in, out, err);
    Wolf w = new Wolf("Meat");

    try {
        w = w.read(aio);
    } catch (IOException e) {
        fail("read() failed for Wolf");
        e.printStackTrace();
    }
    assertEquals("Wolf howls, eats test", w.toString());
    System.out.println("readWolf returned: " + w);
}
```

## Output

Console Problems Javadoc Decl

```
<terminated> TestWolf [JUnit] /Library/Java/JavaVirtual
printWolf wrote: Wolf howls, eats Meat
readWolf returned: Wolf howls, eats test
|
```

# Updating Pack

# Tasks to update Pack

- Add testPrint to TestPack
- Add print() to Wolf
- Run testPrint()
- Add testRead to TestWolf
- Add read() to Wolf
- Run testRead()

# Add testPrint to TestPack

Create Streams

Create AnimalsInOut

Add wolves to Pack

```
@Test
public void testPrintPack() {
    BufferedReader in = new BufferedReader(new StringReader("test"));
    ByteArrayOutputStream outString = new ByteArrayOutputStream();
    PrintStream out = new PrintStream(outString);
    ByteArrayOutputStream errString = new ByteArrayOutputStream();
    PrintStream err = new PrintStream(errString);

    AnimalsInOut aio = new AnimalsInOut(in, out, err);
    Pack p = new Pack();

    p.addWolf(new Wolf("Meat"));
    p.addWolf(new Wolf("Meat"));
    p.addWolf(new Wolf("Meat"));
    p.print(aio);
    // ...
}

// ...

@Test
public void ...
    Bu
    ou

    ai
    Pa

    p
```

as Create method 'print(AnimalsInOut)' in type 'Pack'...

Sy () Add cast to 'p'

Rename in file (%2 R)

```
return "Pack contains " + members.size() + " wolves";
}

public void print(AnimalsInOut aio) {
    // TODO Auto-generated method stub
}

}
```

Press 'Tab' from proposal table or click for focus

# Add print() to Pack

```
public void print(AnimalsInOut aio) {  
    aio.getOut().print("Pack contains " + members.size() + " wolves");  
}
```



# Test print() in Pack

▼ animals.TestPack [Runner: JUnit 4] (0.000 s)

testAddWolf (0.000 s)

testHunts (0.000 s)

testPrintPack (0.000 s)

testConstructor (0.000 s)

@Test

```
public void testPrintPack() {
    BufferedReader in = new BufferedReader(new StringReader("test"));
    ByteArrayOutputStream outString = new ByteArrayOutputStream();
    PrintStream out = new PrintStream(outString);
    ByteArrayOutputStream errString = new ByteArrayOutputStream();
    PrintStream err = new PrintStream(errString);

    AnimalsInOut aio = new AnimalsInOut(in, out, err);
    Pack p = new Pack();

    p.addWolf(new Wolf("Meat"));
    p.addWolf(new Wolf("Meat"));
    p.addWolf(new Wolf("Meat"));
    p.print(aio);
    assertEquals("Pack contains 3 wolves", outString.toString());
    System.out.println("testPrintPack() returned: " + outString.toString());
}
```

Output

Console Problems Javadoc Declaration

<terminated> TestPack [JUnit] /Library/Java/JavaVirtualMachines  
testPrintPack() returned: Pack contains 3 wolves

# Add testRead to TestPack

```
@Test
public void testReadPack() {
    BufferedReader in = new BufferedReader(new StringReader("Deer\nDeer\nMeat"));
    ByteArrayOutputStream outString = new ByteArrayOutputStream();
    PrintStream out = new PrintStream(outString);
    ByteArrayOutputStream errString = new ByteArrayOutputStream();
    PrintStream err = new PrintStream(errString);

    AnimalsInOut aio = new AnimalsInOut(in, out, err);
    Pack p = new Pack();
```

```
    p = p.read(aio);
```

```
    assert
```

```
    System
```

```
    }
```

```
}
```

e Prob

d> TestPack [J

tPack() ret

Create method 'read(AnimalsInOut)' in type 'Pack...'  
Add cast to 'p'  
Rename in file (%2 R)

```
    aio.getOut().print("Pack contains " + members.size() +  
        " wolves");  
}
```

```
public Pack read(AnimalsInOut aio) {  
    // TODO Auto-generated method stub  
    return null;  
}
```

Press 'Tab' from proposal table or click for focus

# Add read(AnimalsInOut aio) to Pack

```
public Pack read(AnimalsInOut aio) {  
    String food = null;  
    Scanner s = null;  
  
    try {  
        s = new Scanner(aio.getIn());  
  
        while (s.hasNext()) {  
            food = s.next();  
            this.addWolf(new Wolf(food));  
        }  
    } finally {  
        s.close();  
    }  
    return this;  
}
```

# Run Test

▼ animals.TestPack [Runner: JUnit 4] (0.001 s)

- testAddWolf (0.001 s)
- testHunts (0.000 s)
- testPrintPack (0.000 s)
- testReadPack (0.000 s)
- testConstructor (0.000 s)

```
public void testReadPack() {  
    BufferedReader in = new BufferedReader(new StringReader("Deer\nDeer\nMeat"));  
    ByteArrayOutputStream outString = new ByteArrayOutputStream();  
    PrintStream out = new PrintStream(outString);  
    ByteArrayOutputStream w = new ByteArrayOutputStream();  
    PrintStream rString);  
  
    AnimalsInOut aio = new AnimalsInOut(in, out, err);  
    Pack p = new Pack();  
  
    p = p.read(aio);  
    assertEquals("Pack contains 3 wolves", p.toString());  
    System.out.println("readPack returned: " + p);  
}
```

Output

Console Problems Javadoc Declaration

```
<terminated> TestPack [JUnit] /Library/Java/JavaVirtualMachines  
testPrintPack() returned: Pack contains 3 wolves  
readPack returned: Pack contains 3 wolves
```