

**1.00 - Introduction to Computation and
Problem Solving**

**Class 9:
Static Methods and Data Members**

**Prof. Steven R. Lerman
and
Dr. V. Judson Harward**

Announcements/Reminders

- **The first quiz will be during class, 11:00 on Friday, October 7. The quiz is open book/open notes, but no electronic devices are allowed.**
- **We will schedule at least one review session during the week before the quiz.**
- **My own office hours this week are cancelled.**

Goals

- This the session in which we explain what static means.
- You will learn how to write classes with static methods and data members.

3

The UnitCircle Program

```
public class UnitCircle {
    public static void main( String [] args ) {
        double circum = 2 * Math.PI;
        System.out.println(
            "Circumference of the unit circle = "
            + circum );
    }
}
```

4

Math.PI

```
double circum = 2 * Math.PI;
```

- **What is Math? Class, object, or method? How do you know? Can you create Math instances?**
- **What is PI? Class, data member, or method? It's defined as**

```
public static final double PI =  
    3.14159265358979323846;
```

– **final** makes it “read only”

5

Why Math.PI is static data member

- If **PI** is a data member, what object does it belong to?
- We can't create **Math** instances so it can't belong to an instance of the **Math** class.
- Because **PI** is declared static, it belongs to the **Math** class object . Access it using the class name, not an object reference.
- Yes, even classes are objects.
- How many **Math** class objects are there in any Java program? How many **PI** data members?

6

`System.out.println()`

```
System.out.println( "Circumference= "  
    + circum );
```

- What is `System`? Class, object or method?
- Can you create an instance of `System`?
- What is `out`? Class, object, data member, method?
- What is `println()`? In what class do you think `println()` is declared?
- How do you think `System.out` is declared?

7

`System.out`

```
public final static PrintStream out;
```

- `public` so that we can print to it from any object
- `final` so that the user can't replace it
 - You can not replace a final object reference, but you can modify the object referred to. Every time we call `println()` we modify `out`.
- `static` so that there will be exactly one output connection to the user's terminal window
- `PrintStream` because that class has all the right methods

8

Classes Without Instances (Preview)

How do you write a class that can not have instances?

- Declare it to be abstract

```
public abstract class NoInstanceClass  
{ ... }
```

9

Classes Without Instances (Preview)

- Make the default constructor private

```
public class NoInstanceClass {  
    private NoInstanceClass() {}  
}
```

- If a class has no constructor, Java will supply a default constructor
- But if you supply any constructor, then Java won't create one.
- If you define only one constructor and make it private, you can't create an instance outside the class. (You can "new" an instance inside a class with a private constructor.)

10

When Do static Members get Initialized?

- Just before they are used. In general the compiler takes care of this.
- But the compiler can't resolve paradox:

```
public class Init { // Evil
    public static final int i1 = 2*i2;
    public static final int i2 = 2*i1;
}
```

11

static Colors

- Java has a `Color` class in `java.awt` package that we will use extensively when we create graphic user interfaces.
- The usual way to create a new `Color` is to use a constructor that takes a red, green and blue component:

```
Color tangerine = new Color( 255, 100, 50 );
```

- `Color` also defines certain frequently used colors as static so you don't need to create them. For instance, `Color.RED` is defined as

```
public final static Color RED =
    new Color(255, 0, 0);
```

12

Summary of `static` Data Members

- **Belong to the class, not an instance of the class so there is only one copy**
- **Typically `public` so you can access them using `MyClass.myStatic`**
- **Often `final` so they can not be “changed”**
- **Typical uses:**
 - **Defined constants: `Math.PI`, `Color.RED`**
 - **Fixed resources: `System.out`**
 - **Class-wide values such as counters of instances**

13

`static` Methods

- **Most methods are called on an instance of a particular class.**
- **`static` methods are not called on an instance. Instead they are conceptually called on the class.**
- **Methods in `Math` package (e.g. `Math.pow()`) are good examples.**

14

Example Methods

- **Why does the Java entry point method**
`public static void main(String [] args)`
have to be static?
- **Why is `Math.pow(double a, double b)`**
static?
- **Why is `int Integer.parseInt(String s)`**
static?
- **Why isn't the `int intValue()` method in**
`Integer` static?
- **Why isn't `System.out.println(String s)`**
static?

15

The Syntax of static Methods

- **Defining:**
`// static`
`public static int parseInt(String s)`
`// instance`
`public int intValue();`
- **Calling:**
`int ival = Integer.parseInt("42");`
`Integer bigI = new Integer("42");`
`ival = bigI.intValue();`

16

static Methods Can't Access Instance Data or Methods

Find the errors (2):

```
public class StaticError {
    private int k;

    public StaticError() { k = 42; }

    public int getValue() { return k; }

    public static void main( String [] args ) {
        int k2 = 2*k;
        System.out.println( "k= " + getValue() );
    }
}
```

17

Instance or static as a Matter of Style

- Sometimes it is just a matter of style whether you declare a method to be an instance or a static method.
- Consider a class `Work` used for physics and engineering calculations. The constructor takes 2 arguments, a `double` and a `String` indicating the units of the first argument.
- Remember that a confusion on this issue between NASA and a contractor led to the loss of a Mars probe a few years ago.

18

class Work

```
public class Work {  
    public static final String JOULE = "JOULES";  
    public static final String LBF = "LBF";  
    public static final String BTU = "BTUS";  
    ...  
  
    private double val;    // in joules  
  
    public Work( double w, String units )  
    { ... }  
}
```

19

Instance or static, 2

Which is the better way to add two instances of work?

```
public static Work add(Work w1, Work w2);
```

or

```
public Work add( Work w );
```

20

Instance or static?

- In the Work class,

```
// returns the value in joules  
double toJoules();
```

Instance

```
Work w = new Work( 4.3, Work.LBF );  
double j = w.toJoules();
```

21

Instance or static?, 2

- In the Work class,

```
// returns the conversion factor from  
// unit u1 to unit u2  
double getConversion( String u1,  
                      String u2 );
```

Static

```
double cf = Work.getConversion(  
    Work.JOULES, Work.LBF );
```

22

Instance or static?, 3

- In the Arrays class,

```
// fill the array with val  
void fill( int [] a, int val);
```

Static:

```
int [] a = new int[ 10 ];  
void Arrays.fill( a, 42 );
```

23

Instance or static?, 4

- In the Date class,

```
// takes a String date,  
// returns milliseconds since 1/1/70  
long parse( String s );
```

Static:

```
long l = Date.parse(  
    "9/24/2002" );  
Date d = new Date( l );
```

24

Instance or static?, 5

- In the newer `DateFormat` class, which represents a particular way of formatting dates, e.g., US 9/24/2002 vs. European 24/9/2002

```
// takes a String date,  
    Date parse( String s );  
Instance:  
// gets default local DateFormat  
DateFormat df =  
    DateFormat.getDateInstance();  
Date d = df.parse( "9/24/2002" );
```

25

Static Members Exercise

- Using your browser, navigate to the lecture notes section and download `Lecture9JavaFiles.zip`.
- Double click on the file you downloaded and extract it. It will unpack into a new subdirectory call `Lecture9JavaFiles`.
- You will see a single Java file and a `README.pdf` file that has instructions for this exercise.

26