

---

# Static Methods and Fields

CSE 142, Summer 2002  
Computer Programming 1

<http://www.cs.washington.edu/education/courses/142/02su/>

# Readings and References

---

- Reading
- Other References
  - » Sections 8.3.1.1 static Fields and 8.4.3.2 static Methods, *Java Language Specification*, Second Edition

# Method main

---

- We need to identify the point where program execution starts
- The Java convention is that we define a static method named **main** in at least one class

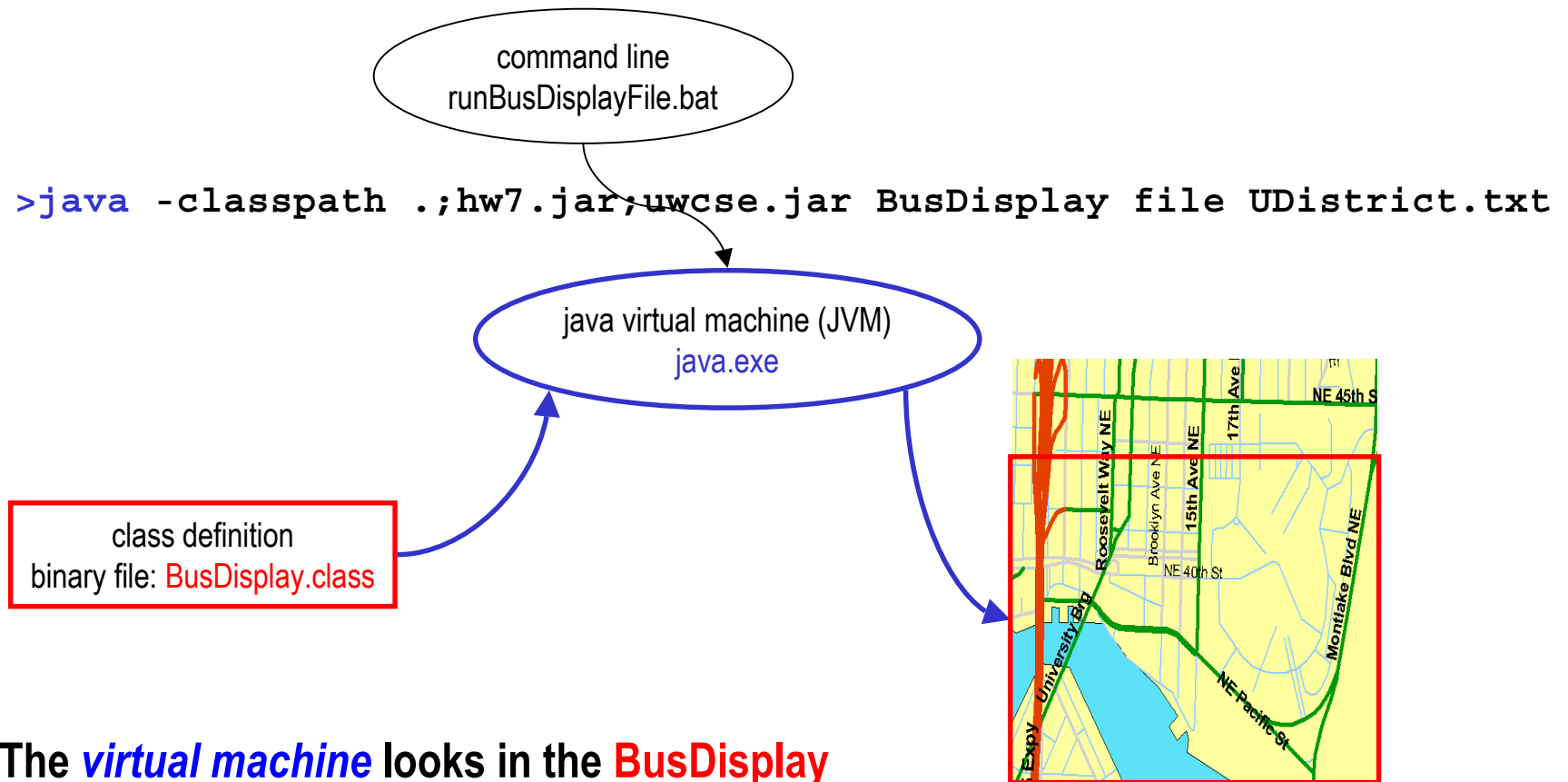
```
public static void main(String[] args) {...}
```
- Then we start the java virtual machine running and tell it the name of the class that contains the **main** method that we want to use

# Launching a java program

---

- The java tool launches a Java application. It does this by
  - » starting a Java runtime environment
  - » loading a specified class
  - » invoking that class' main method.
- The method declaration must look like the following:
  - » `public static void main(String args[])`

# Running a Java program



# Method main

---

- » "static" is used to identify methods and variables that exist automatically *before any objects are created*
- » main must always be defined like this

```
public static void main(String[ ] args) { ... }
```
- » Typical contents of main
  - create some objects and call some methods to get started
- » args array contains any string arguments passed to the program when it was started.
  - Actual name need not be "args"

# method main in BusDisplay

---

```
/**
 * This class manages the process of reading bus events and displaying them.
 */
public class BusDisplay {
    /**
     * This program can read bus event records from a file or from a network
     * port.
     */
    public static void main(String[] arg) throws IOException {
        /* open the requested input stream */
        ...
        /* build the map display window */
        ...
        /* read events and add them to the map */
        ...
    }
}
```

# Static methods and fields

---

- Static members belong to the class, not to any instance of the class
  - » Both fields and methods can be static
  - » Java VM creates memory for static fields when the class is first used
- Static methods are often “utility” methods that don’t need an instance of the class
- Static fields are sometimes used as constants



# static methods

---

- Recall that methods implement behavior
  - » but some behaviors are not associated with a specific object for one reason or another
  - » static methods are often used as “utility” methods
- Static methods must be implemented in a class
  - » There are some “pure” utility classes like `java.lang.Math` and `java.lang.System`
  - » Other classes have some utility methods along with instance methods, like `Integer.toString(int k)`

# java.lang.Math

---

```
static double abs(double a)
static double ceil(double a)
static double floor(double a)
static double rint(double a)
static long round(double a)
static int round(float a)

static double acos(double a)
static double asin(double a)
static double atan(double a)
static double atan2(double a, double b)
static double cos(double a)
static double sin(double a)
static double tan(double a)
static double toDegrees(double angrad)
static double toRadians(double angdeg)

static double exp(double a)
static double IEEERemainder(double f1, double f2)
static double log(double a)
static double pow(double a, double b)
static double random()
static double sqrt(double a)
static double max(double a, double b)
static double min(double a, double b)
```

```
double deg = Math.toDegrees(radians);
double distance = Math.sqrt(dx*dx + dy*dy);
```

# java.lang.System static methods

---

`static long currentTimeMillis()`

Returns the current time in milliseconds.

`static void exit(int status)`

Terminates the currently running Java Virtual Machine.

`static Properties getProperties()`

`static String getProperty(String key)`

`static String getProperty(String key, String def)`

`static void setProperties(Properties props)`

`static String setProperty(String key, String value)`

Read and write the current system properties.

`static void setErr(PrintStream err)`

`static void setIn(InputStream in)`

`static void setOut(PrintStream out)`

Reassign the standard input, output, and error streams

# System.exit(int status)

---

```
try {  
    ... attempt to initialize ...  
}  
catch (NumberFormatException e) {  
    System.out.println(usage);  
    System.exit(1);  
}  
catch (IOException e) {  
    System.out.println(e);  
    System.exit(1);  
}
```

# More examples of static methods

---

**String Integer.toString(int i)**

- » Returns a new String object representing the specified integer

**int Integer.parseInt(String s)**

- » Parses the string argument as a signed decimal integer.

**void Collections.sort(List list)**

- » Sorts the specified list into ascending order, according to the natural ordering of its elements.

**InetAddress InetAddress.getByName(String host)**

- » Determines the IP address of a host, given the host's name.

# Static Fields

---

- Recall that instance variables are created fresh for each instance of the class
  - » each instance has its own separate instance variable
- Static variables are only created once
  - » all references to a static variable point to exactly the same place in memory
- Static variables can be used to keep global values
  - » how many objects of some class have been created
  - » how many streams are open to another host machine

# Static Field Example

---

Only one copy of a static member exists for a class

```
public class Employee {  
    public Employee ( int salary ) {  
        baseSalary = salary;  
        numberOfInstances++;  
    }  
    static private int numberOfInstances = 0;  
    private int baseSalary;  
}
```

# Constants: Static Final Fields

---

- Sometimes we just want to give a name to a constant value, like PI or E
- Solution: a static variable, but further qualified with *final* so it can't be changed after it is initialized.

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

- Final variables must be initialized when declared
  - » cannot be changed later



# Constants in the Java Libraries

---

- Class Math contains PI and E, with the expected values.
  - » `this.area = Math.PI * this.radius * this.radius;`
- Classes like Integer and Double contain values like the largest possible int value, the error value NaN (not a number), etc
  - » `int biggest = Integer.MAX_VALUE;`
  - » `double val = Double.NaN;`

# More useful constants

---

- The `java.awt.Color` class contains some preset color definitions

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

```
» Color bg = java.awt.Color.white;      // java 1.3
```

```
» Color bg = java.awt.Color.WHITE;     // java 1.4
```

- The `java.lang.System` class contains some preset input / output streams
  - » `System.out.println("hi there");`