Debugging

- Primitive numerical types
  – Shorthand assignment statements
  – Type conversions
- Using Javadocs
- Using jdb - the Java debugger
  – Commands

Primitive Numerical Types

- **“Shorthand” assignments**

  <assign-stmt> $\rightarrow$ **<var >** ++ ***//increment by 1***

  <assign-stmt> $\rightarrow$ **<var> --** ***//decrement by 1***

  <assign-stmt> $\rightarrow$ <**var**> += **<expr>** ***//incr by <expr>***

  <assign-stmt> $\rightarrow$ <**var**> –= <**expr**> ***//decr by <expr> value***

- **Synonyms**

  `i++` **and** **`i+=1`**

Type Conversions
Primitive Numerical Types

- *Widening* - a value conversion without loss of precision

  int $\rightarrow$ `double`, `long` $\rightarrow$ `double`

- *Narrowing* - a value conversion with possible loss of precision - needs a type cast

  double $\rightarrow$ int

  `(int)6.6` yields `6`

  To obtain a rounded value must use Math.round()

  class method invocation: <classname>.<method_name>

Using Javadocs

- Java runtime system contains many standard packages which you can use (import) in your Java programs
- Click on Java Development Kit Packages on cs111 Java Documentation
- Interesting Java API Packages:

  package java.applet

  package java.awt

  package java.beans

  package java.io

  package java.lang

  package java.math

  package java.util

java.lang Package Webpage

- Lists interfaces, classes, exceptions and errors associated with this package

package java.lang

Boolean

Byte

Character

Math

...

Math Class in java.lang

```
Class java.lang.Math
java.lang.Object
   |
   +----java.lang.Math
public final class Math
extends Object
```

class relationships

The class Math contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

class description

Math Class Details

- **Lists variables and methods of class with signatures, followed by additional info**
- **abs(double)**

  Returns the absolute value of a double value.

public static double abs(double a)

Returns the absolute value of a double value. If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned.

Parameters:

a - a double value.

Returns:

the absolute value of the argument.

```
Math.abs(3.0)
```

Debugging with jdb

- Used UStime-procs-err.java to seed an error in UStime-procs.java and show how jdb works.

- Also in main method added new method invocation:

```
String t = null;
System.out.println((z.timeConvert(t))
    + "in San Francisco \n");
```

Error-seeded Method

UStime-procs-err.java

```
public UStime timeConvert(String s) {
  int d, timeDiff;
  UStime t = new UStime(0,0);
  if (s == "Pacific") timeDiff = 3;
      else if (s == "Mountain") timeDiff = 2;
      else if (s == "Central")  timeDiff = 1;
      else
  {System.out.println("Error in input time zone given" +
      s.toString());
      System.exit(1);
      timeDiff = 0;}
  t.hours = (this.hours + 12 - timeDiff) % 12;
  if (t.hours == 0)  t.hours = 12;
  t.minutes = this.minutes;
  return t;
}
```

added an unnecessary call to toString() on a String object

How to run jdb?

```
22 remus!111> javac -g UStime-procs-err.java
23 remus!111> jdb
Initializing jdb...
> run UStime
running ...
```

java program

class with main method to run

jdb output

```
running ...
main[1] Twelve noon EST is 11 hours and 0 minutes in
  Chicago
Twelve noon  EST is 10 hours and 0 minutes in Denver
Twelve noon  EST is 9 hours and 0 minutes in San
  Francisco
Uncaught exception: java.lang.NullPointerException
  at UStime.timeConvert(UStime-procs-err.java:23)
  at UStime.main(UStime-procs-err.java:51)
  at sun.tools.debug.MainThread.run(Agent.java:55)
main[1]
```

call chain of trace

list

```
main[1] list
19 else if (s == "Mountain") timeDiff = 2;
20 else if (s == "Central")  timeDiff = 1;
21 else
22 {System.out.println("Error in input time zone given"
   +
23       =>                    s.toString());
24                             System.exit(1);
25                             timeDiff = 0;}
26 t.hours = (this.hours + 12 - timeDiff) % 12;
27 if (t.hours == 0)  t.hours = 12;
main[1]
```

locals

```
main[1] locals
Method arguments:
  this = 12 hours and 0 minutes
  s = null
Local variables:
  timeDiff is not in scope
  t = 0 hours and 0 minutes
  timeDiff is not in scope
  timeDiff is not in scope
  timeDiff is not in scope
```

indicates timeDiff has not yet been initialized on this execution path

Navigating the call chain

```
main[1] where
  [1] UStime.timeConvert (UStime:23)
  [2] UStime.main (UStime:51)
  [3] sun.tools.debug.MainThread.run (MainThread:55)
main[1] up
main[2] locals
Method arguments:
  args =
Local variables:
  z = 12 hours and 0 minutes
  t = null
```

error occurred in timeConvert()
look at its caller

Finding the call site in main

```
main[2] list
47                  "in San Francisco" + "\n");
48         //   System.out.println((z.timeConvert("Alaska")) +
49         //           "in Alaska");
50              String t = null;
51      =>        System.out.println((z.timeConvert(t)) +
52                  "in San Francisco \n");
53         }
54
55         }
```

Examining Values

```
main[2] print  z
z = 12 hours and 0 minutes
main[2] dump z
z = (UStime)0xee32b210 {
    private int hours = 12
    private int minutes = 0
}
main[2] print t
"t" is not a valid local or class name.
```

variable or object name

hint at source of error

More Navigation

```
main[2] down
main[1] print this
this = 12 hours and 0 minutes
main[1] dump this
this = (UStime)0xee32b210 {
    private int hours = 12
    private int minutes = 0
}
main[1] print s
"s" is not a valid local, class name, or field of (UStime)0xee32b210
main[1] exit
24 remus!111>
```

graceful end of jdb session