W1004 Programming Review Session 6

12/9/2011 - Don Yu

File IO

- One way is to make a File object and pass that to PrintWriter and Scanner class for IO
- Then you can use all the methods that you used before with Scanner and System.out.
- Remember to call close() after you finish writing to a file
- Remember to catch FileNotFoundExceptions that occurs
- Command line arguments are accessed through the array args
  - If I give a command line argument such as
    - java ExecuteTest -d -p input.txt boo
    - args[0] = "-d"
    - args[1] = "-p"
    - args[2] = "input.txt"
    - args[3] = "boo"

Exceptions

- You throw an exception using the "throw" statement
- You tell a method that it might throw this exception with the "throws" statement
- ^^Know the difference in syntax
- A checked exception : one in which the compiler will make sure you don't forget about it
  - Examples are FileNotFoundException for when you read/write to a File
- An unchecked exception : one in which you are on your own to throw/catch it
  - Examples are ArrayIndexOutOfBounds
- Finally clause is ALWAYS run

Example Final Questions

- Which of the following exceptions are checked exceptions?
  - NullPointerException
  - NoSuchElementException
  - IndexOutOfBoundsException
  - FileNotFoundException
- What does the following code print out if the file does not exist?
  - ```
    try {
        File f = new File("input.txt");
        Scanner s = new Scanner(f);
        s.nextLine();
    } catch(Exception e) {
        System.out.println("bad things happened");
    } finally {
        System.out.println("later");
    }
    ```
    ```

Inheritance

- Defines an is-a relationship (keyword is "extends")
- Example: If the class Car extends Vehicle then it will have all the methods of that class
- The subclass Car can override methods of superclass Vehicle
- The keyword "super" is used to access superclass methods and constructors; ex -> super(); will call constructor

REMEMBER

- Inheritance is transitive (If B extends A and C extends B then C also extends A)
- Private Members of the superclass are NOT inherited by the subclass and have to be accessed indirectly
- In Java you can only extend (inherit) from one other class

Interfaces

- Think of this as a contract with a class; if you implement my Interface then you must include these methods
- Example: If you have you have your class implement Comparable
  - public class Card implements Comparable {

```
.....
//must implement this
public int compareTo(Card c) {....}
}
```

REMEMBER

- An interface does not provide any implementation for any methods
- All methods in an interface type are automatically public
- An interface does not have instance variables

Polymorphism

- Literally means "many forms"
- This is a very abstract concept so let's do an example
- Say I have the following code
  - Animal[] animals = new Animal[];
  - animals[0] = new Human();
  - animals[1] = new Animal();
  - animals[2] = new Dolphin();
- Which method do I call when I write this?
  - animals[0].breathe()
- What if Animal is an interface not a class? Is there an error now?
- Can we even have an array of Animal if its just an interface??

Answers

- Because animals[0] is in fact a Human object it will call the breathe() method of the Human class! (even if it was declared as an Animal)
- This is because Java will actually look at what the actual class of the object is and call the corresponding method! It doesn't matter what you declare it as (must be initialized with a class that extends or implements the declared type)
- Yes you CAN NOT create an object out of an interface so new Animal() would be an error
- Yes you can have an array of type interfaces! As long as you set the elements of that array to objects that implement the interface then it's okay. The Java virtual machine will determine what class the actual object is! It's magical

Example Code

- Let's go over a simple Java program that emphasizes greedy algorithms
- Greedy Algorithm : Pick the best choice at each step
- We have already seen a very important greedy algorithm in W1004 already
- Djikstra's is a greedy algorithm - pick the node that has the shortest path and analyze that at each step

Brainstorm

- What classes should I have? (at the minimum)
- What exceptions should I catch?
- What should each class do?
- What should my output be?
- Remember it is best to go about any assignment one step at a time

Specifics about the assignment?

- You must read from a file the name of the talk, the start time, and the end time
- You can assume that the format will be like what's stated in the assignment
- For output you can either print to standard out or you can output to a new File (but be sure to mention this in your readMe!!)

Things to watch out for

- Your program must (ABSOLUTELY MUST) compile and run without error

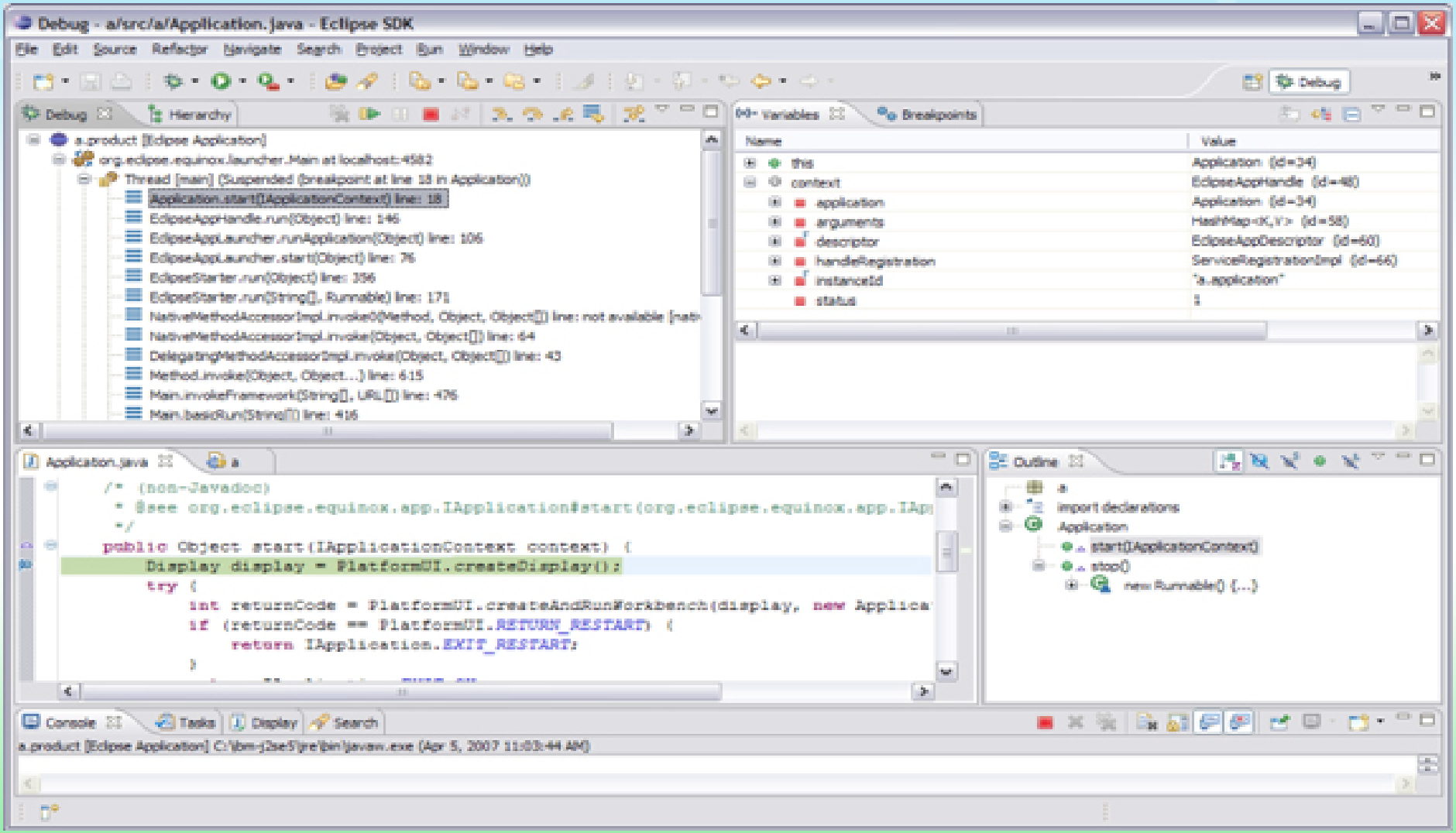
Debugging

- Eclipse has a very powerful debugger prebuilt into the IDE
  - Step 1 : Set Breakpoints
  - Step 2 : Run Debugger
  - Step 3 : Click Play (if your breakpoint is not hit yet)
  - Step 4 : Roll mouse over variables to see what they are at this moment in time
  - Step 5 : Set other breakpoints and repeat until you figure out what the problem is
- This is a very basic tutorial and does not cover all that the debugger has to offer
- Great to figure out when objects are null, if you're ever reaching a part of your code, what your objects contain at any moment

1 : Set Breakpoints by double-clicking on left side (will show up as blue dots) *double click again to get rid of it

```
    /* (non-Javadoc)
     * @see org.eclipse.equinox.app.IApplication#start(org.eclipse.equinox.app.IApplicatic
     */
    public Object start(IApplicationContext context) {
        Display display = PlatformUI.createDisplay();
        try {
            int returnCode = PlatformUI.createAndRunWorkbench(display, new ApplicationWork
            if (returnCode == PlatformUI.RETURN_RESTART) {
                return IApplication.EXIT_RESTART;
            }
            return IApplication.EXIT_OK;
```

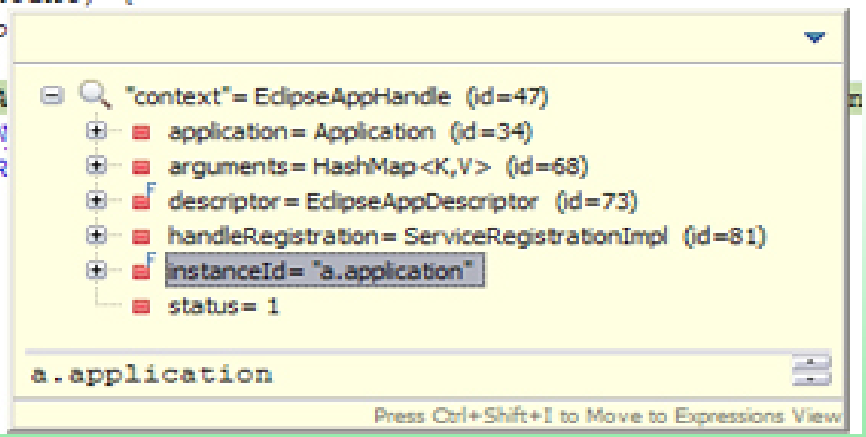
Click Debug button (looks like an ant next to the play button) to go into debug view



Inspect your variables when your breakpoint is hit by rolling over them, right-clicking and then going to inspect, or with ctrl-shift-i (shift-cmd-i on macs)

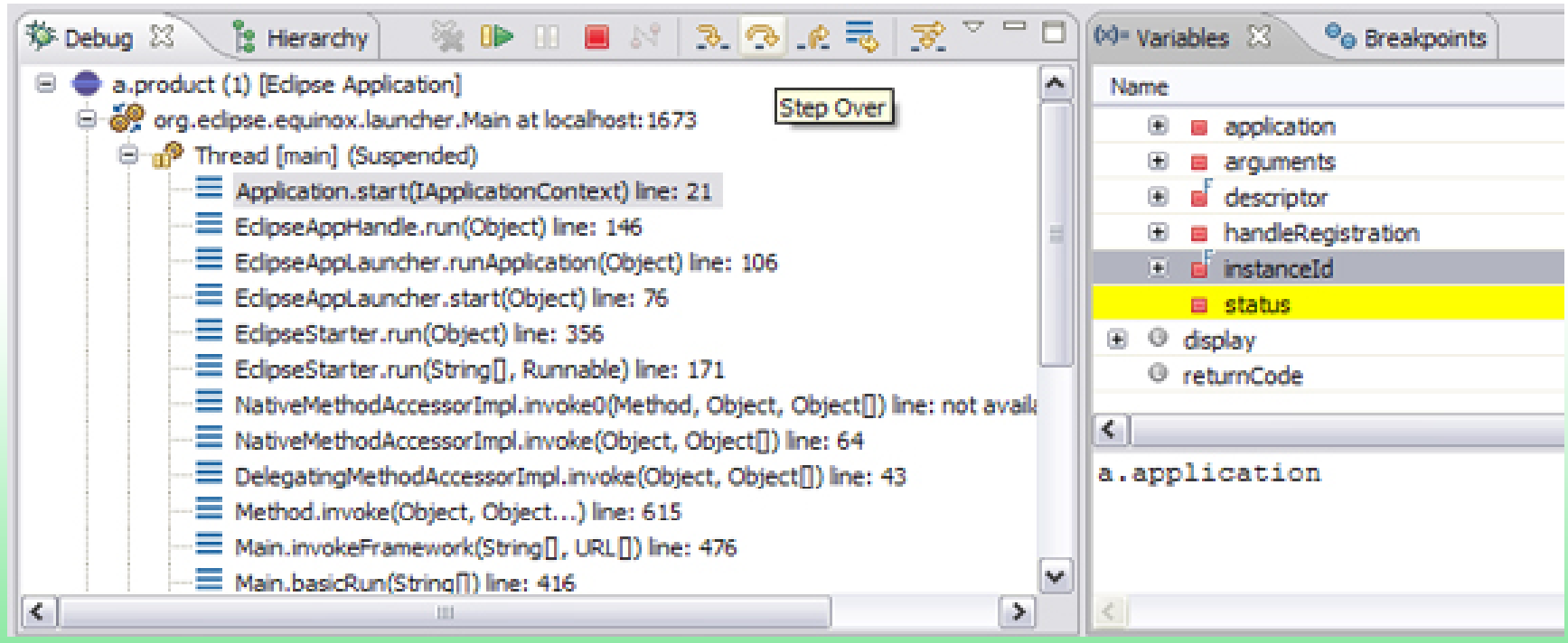
```
    /* (non-Javadoc)
     * @see org.eclipse.equinox.app.IApplication#start(org.eclipse.equi
     */
    public Object start(IApplicationContext context) {
        Display display = PlatformUI.createDisplay();
        try {
            int returnCode = PlatformUI.createAndRunWorkbench(display,
            if (returnCode == PlatformUI.RETURN_RESTART) {
                return IApplication.EXIT_RESTART;
            }
```

```
/* (non-Javadoc)
 * @see org.eclipse.equinox.app.IApplication#start(org.eclipse.equinox.app.IApplicationCo
 */
public Object start(IApplicationContext context) {
    Display display = PlatformUI.createDisp
    try {
        int returnCode = PlatformUI.createA
        if (returnCode == PlatformUI.RETURN
            return IApplication.EXIT_RESTAR
        }
        return IApplication.EXIT_OK;
    } finally {
        display.dispose();
    }
}
```



Keep stepping through your code one step at a time (inspecting variables along the way)

The Step Over button (highlighted in picture below) will go to the next line of code that is executed in your program. Stepping in button (to the left of step over) will enter into a method!



Repeat until your program works!

- What we didn't cover
  - Conditional Breakpoints
  - Suspending Threads
  - Remote debugging
  - On the fly code fixing
  - Debugging other languages (yes Eclipse can do more than just Java!)
  - Much more

Final Review

- Same format as the midterm (but longer obviously)
  - Multiple Choice
  - Definitions
  - Short Answer
  - Debugging
- Make sure you know djikstra, turing machines, halting problem, different sorting algorithms, Java IO, exceptions, inheritance, polymorphism, network layers/protocols, etc.
- Will also have questions from the first half of the course so don't neglect that
- Practice debugging with your friends

Questions??