Software Construction Anonymous

Dhammika Elkaduwe

Department of Computer Engineering
Faculty of Engineering

University of Peradeniya

ILOs

- Anonymous classes
- Threads using anonymous classes

Basic idea: Anonymous

Meaning: without a name

What: anonymous classes and functions

Why: classes that will be used once

Syntax: example 1

see Ex1.java

```
public class Ex1 {

  interface SaySomething {
    public void greet();
  }
  ...
```

Things to note:

- the *interface* is defined within the class
- the interface cannot be used in other places (outside of this class)
- see what happens after compilation

Syntax: example 1

see Ex1.java

```
public class Ex1 {
  public static void main(String [] args) {
    class SayHi implements SaySomething {
      String whatToSay = "Hi world";
      // no constructor. Default will be used
      public void greet() {
        System.out.println(whatToSay);
      }
    }// end of class definition
    SaySomething hi = new SayHi();
```

Things to note:

- class is defined within the main function
- Not really anonymous (since it has a name)
- But cannot be used outside the *Ex1* class
- If no constructor is given the default will be used

Syntax: example 1

see Ex1.java

```
public class Ex1 {
  public static void main(String [] args) {
    SaySomething bye = new SaySomething() {
      // class is defined here.
      // cannot be used in another place
      String whatToSay = "bye world";
      public void greet() {
        System.out.println(whatToSay);
      }
    };// end of class
    hi.greet();
    bye.greet();
```

Things to note:

- Example of an anonymous class
- Class has no name (so obviously cannot use it outside)
- Useful for simple operations

Rules about anonymous classes

see Ex1.java

```
SaySomething bye = new SaySomething() {
  // class is defined here.
  // functions etc goes here
};// end of class
bye.greet();
```

Things to note:

- The *new* operator is there before the definition,
- Followed by a name of interface to implement or class to extend,
- Followed by *()* which cannot take arguments
- Within the body you have definition of functions etc.

Syntax : example 2

- Recall how the *runnable* interface is used when creating threads
- Pass a *runnable* object to the *Thread* construnctor which will return a *Thread* object on which you can call start (join, etc)

see Ex2.java

```
class X implements Runnable {
  ...
  Thread x = new Thread(New X());
```

Syntax : example 2

see Ex2.java

```
static int i;
final static int max = 3;
public static void main(String [] args) {
  for(i=0; i < max; i++) {
    Thread t = new Thread(new Runnable() {
      int id = i;
      public void run() {
        for(int i=0; i < max; i++)
          System.out.println(i + " from thread " + id);
      }});// Thread(
    t.start();
  }// end main
```

- Note how the class is defined within the block

Anonymous classes

Examples:

- Doing threads for fractal computation
- Listing for the mouse click event on the timer
- ...