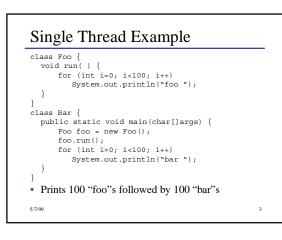


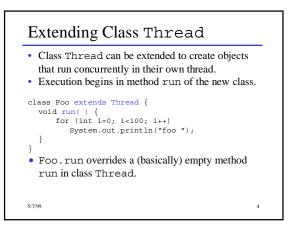
Threads

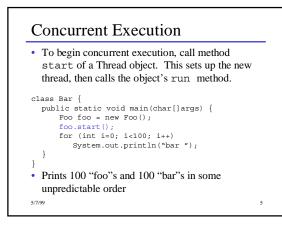
- Thread = Execution of one sequence of instructions (including function/method calls, conditionals, loops).
- Normal Java program executes in a thread created for main (application) or borrowed from the browser (applets).
- Class Thread can be used to create additional threads that execute concurrently.
- Each new thread is associated with (controlled by) a Thread object.

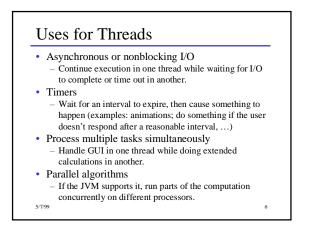
2

5/7/99







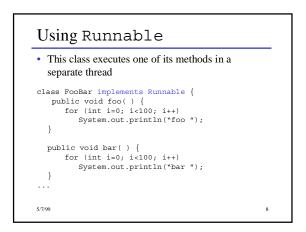


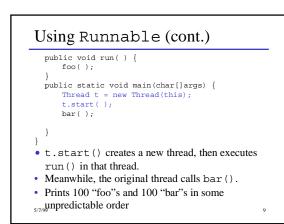
Runnable Classes

- There are many situations where we want to execute a computation concurrently, but in a class that's not a subclass of Thread.
- We still need a Thread object to create and control the thread.
- A thread can begin execution in any class that implements Runnable and contains a run method.

```
public interface Runnable {
    public abstract void run();
}
```

```
5/7/99
```





Synchronization • Since threads may interleave execution in any order, we may need to control access to objects to ensure only one thread at a time can update related variables. class C {

int x,y;
public void setXY(int x, int x) {
 this.x = x; this.y = y;
 public int sumXY() { return x+y; }
}

 What happens if one thread executes sumXY while another thread is halfway through executing setXY on the same object?

synchronized methods Every object has an associated lock We can require threads to acquire the lock before executing one of the object's methods by declaring the method to be synchronized. A synchronized method automatically acquires the object's lock when it is called. Other threads are blocked until the lock is released

acquires the object's lock when it is called. Other threads are blocked until the lock is released automatically when the synchronized method terminates.

5/7/99

11

