

# CSE 11

discussion 8

# Threads, the lazy way

2 ways to make threads:

1. All at once without making classes

Define a new Thread class and the run() method all at once

```
new Thread(){  
    public void run(){  
        //i'm a thread  
    }  
}.start();
```

Similar to what I did in discussion 6 with the Comparator class

2. Make your own class (this is in DrawArc.java which they gave you)

```
public class YoloThread extends Thread{  
    public void run(){  
        //yolo i'm a thread  
    }  
}  
  
new YoloThread().start();
```

**This works for HW8, but extending Thread is really bad in practice**

# Be careful extending Thread

- **There are a lot of methods in Thread you should not override (about 40 of them) or the class won't work the way you expect**
- **start() is one example**

```
public class YoloThread extends Thread{  
    public void run(){  
        System.out.println("I should be running right?");  
    }  
  
    public void start() {  
        System.out.println("YAY I'm starting.");  
    }  
}  
  
new YoloThread().start();
```

Suppose we accidentally override start(). Does this actually launch a Thread?

# Threads, the safer way

- Use the *Runnable* interface!
- Only one method for you to define: `run()`
- Compiler error if you don't define it, so you'll never forget
- No risk of accidentally overriding critical thread functionality

# Threads, the safe way with Runnable

1. All at once without making classes

```
new Thread(new Runnable() {  
    public void run() {  
        //i'm a thread yolo  
    }  
}).start();
```

2. Make your own class

```
public class YoloThread implements Runnable{  
    public void run(){  
        System.out.println("This works!");  
    }  
}
```

```
new Thread(new YoloThread()).start();
```

...OR...

```
Thread t1 = new Thread(new YoloThread());  
t1.start();
```

# Example use for Thread in HW8

Thread 1: get keyboard input

```
while(scnr.hasNext()){  
    //do something with user input  
}
```

Thread 2: update your animation

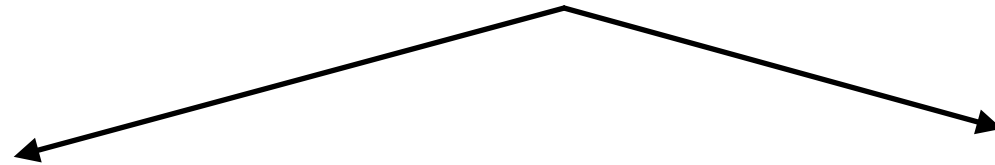
```
while(true){  
    //update animation  
    Thread.sleep(10);  
}
```

# Watch out if 2 threads modify the same thing

(not required for HW8)

ADVANCED

```
ArrayList<String> command_list = new ArrayList<>();
```



Thread 1: writing to list

```
while(scnr.hasNext()){  
    //do something with user input  
    command_list.add(/* something from scn timer */);  
}
```

Thread 2: reading from list

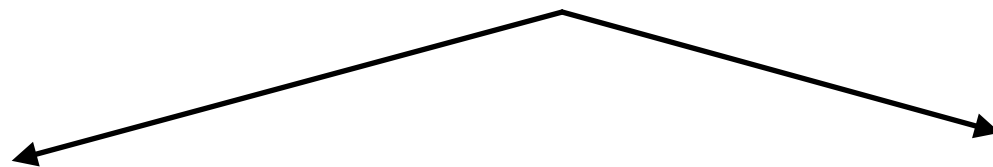
```
while(true){  
    for(String cmd : command_list){  
        //do something with cmd  
    }  
  
    //update animation  
    Thread.sleep(10);  
}
```

**This can give you ConcurrentModificationException**

# Solution: synchronized

ADVANCED

```
ArrayList<String> command_list = new ArrayList<>();
```



Thread 1: writing to list

```
Scanner scnr = new Scanner(System.in);
while(scnr.hasNext()){
    //get something from scanner
    synchronized (command_list) {
        command_list.add(/* something */);
    }
}
```

Thread 2: reading from list

```
while (true) {
    synchronized (command_list) {
        for (String cmd : command_list) {
            System.out.println(cmd);
        }
    }
    //update animation
    Thread.sleep(10);
}
```

synchronized locks an object for exactly 1 thread to use

# HW8: Objectdraw ideas

(demonstration)

- Mouse callbacks. Use this to detect mouse clicks.

```
public class Animate extends WindowController {  
    ...  
    public void onMousePress(Location point){  
        System.out.println(point.getX() + " " +  
point.getY());  
    }  
}
```

- Note that the function signature for “onMousePress” has to match EXACTLY what’s written in the WindowController docs. Otherwise it won’t find your function and you’ll get no mouse press event.

# More objectdraw stuff

- Images:

```
Image img = null;
try {
    img = ImageIO.read(new File("kanye.jpg"));
} catch (IOException e) {
    System.out.println("dude where's kanye");
    System.exit(-1);
}
VisibleImage kanye = new VisibleImage(img, x, y, canvas);
```

- Text:

```
Text t = new Text("i am kanye", x, y, canvas);
```

- `move()` just applies an offset, `move_to()` moves it to a specific place

# demo

(kanye west nbody simulation)

