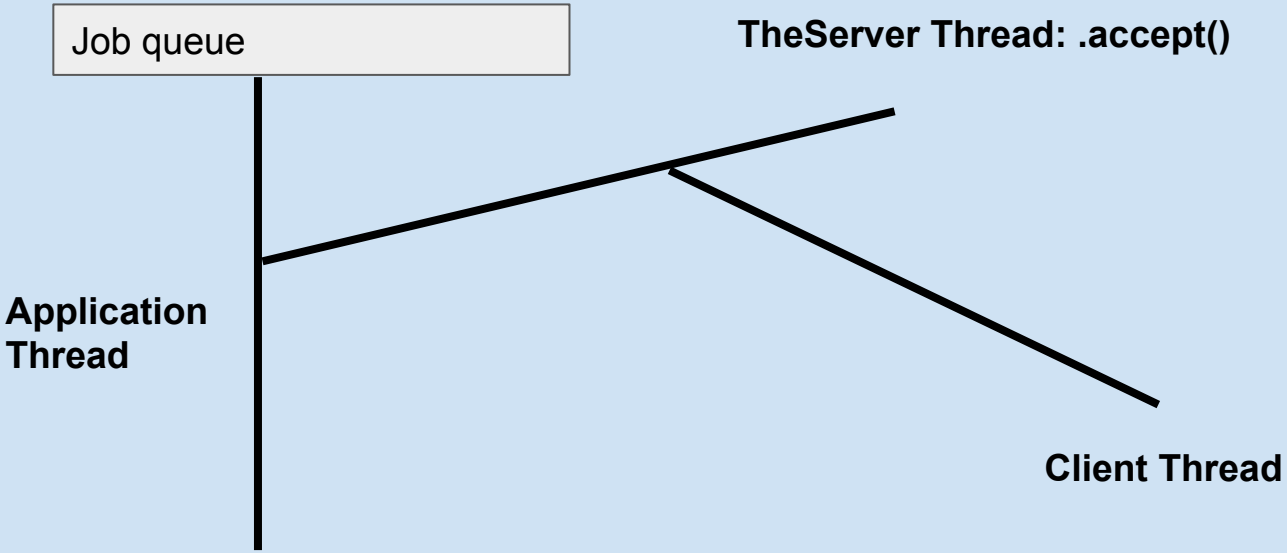
# CS 342 Software Design

- More on Platform.runLater()
- Lambda expressions in Java
- Extra Credit Opportunity

# Platform.runLater() “Threads post jobs on the Application thread job queue”



# Lambda Expressions:

- Enable to treat functionality as a method argument, or code as data.
- A function that can be created without belonging to any class.
- A lambda expression can be passed around as if it was an object and executed on demand.

# Lambda Expressions:

Lambda expressions express instances of functional interfaces (An interface with single abstract).

lambda expressions implement the only abstract function and therefore implement functional interfaces.

# Functional Interface:

A functional interface is any interface that contains only one abstract method. (A functional interface may contain one or more default methods or static methods.) Because a functional interface contains only one abstract method, you can omit the name of that method when you implement it. To do this, instead of using an anonymous class expression, you use a lambda expression.

**Write your own or use one from java.util.function!**

# Lambda Syntax:

- Comma separated list of parameters: (x,y,z)
  - You can omit data type.
  - Omit parentheses if only one parameter
- Arrow: ->
- Body: single expression or a statement block.

Let’s see an example!

# In Class Exercise: Lambda expressions

**This will count for extra credit towards your final grade!**

**Start working on this in groups but everyone must submit their own version to Blackboard.**

**Submit a single .java file with a main() method included.**

**Due: Thursday, April 9th 2020, @11:59**

**No Late submissions!!!**

# Clicker Question: What is this?

```
interface myint{
        static void printstuff() {
            System.out.println("stuff");
        }
        public void op();
        default int mydef() {
            int x = 8;
            int y = 9;
            return x * y;
        }
        public void anotherOp();
    }
```

**A) Nothing, won't compile**

**B) Regular interface**

**C) Functional interface**

**D) I have no idea**

**E) I hate Java!**