The main method

The program's entry point

A program has to start somewhere

You invoke the JVM in order to run a Java application.

- Typically (at least in our books/courses) from the command line
- Using the `java` command followed by the qualified name of the class which contains the special main method
- You can pass along arguments to the main method
  - `java some.package.SomeClassWithMain argument1 argument2 ...`
- In Java, the starting point of an application is the main method
  - a static method with a standardized signature:
    `public static void main(String[] args)`

When main has finished, so has the application

The program only lives as long as the main method is running.

So how come programs don’t end very quickly? Are the main method very long then?

- The main method can contain a loop
- The main method can block (wait) for input (perhaps waiting for user input, or some data on standard in
- The main method typically creates objects and hand over execution to them by calling some method

Main with a loop

```
import java.util.Scanner;

public class AreWeThereYet {
  static Scanner stdin = new Scanner(System.in);
  public static void main(String[] args) {
    while (true) {
      System.out.println("Are we there yet?");
      try {
        String answer = stdin.nextLine();
        if ("yes".equals(answer)) {
          break;
        }
      } catch (Exception e) {
        System.err.println("Problem: " + e.getMessage());
        System.exit(1);
      }
    }
    System.out.println("There.");
  }
}
```

```
$ java AreWeThereYet
Are we there yet?
nope
Are we there yet?
no
Are we there yet?
not yet
Are we there yet?
shut up already
Are we there yet?
yes
There.
```

Main blocking for input

```
import java.io.*;
import java.nio.file.*;

// Counts the lines of stdin or files given as args
public class Jwc {
  public static void main(String[] args) {
    long lines = 0;
    if (args.length > 0) {
      countFile(args);      // reads files
    } else {
      lines = readStdIn();  // blocks in I/O
      System.out.println("Lines: " + lines);
    }
  }
```

```
$ javac Jwc.java && java Jwc *.java
AreWeThereYet.java Lines: 20
HelloFX.java Lines: 42
HelloFXMain.java Lines: 9
HelloMain.java Lines: 13
HelloSayer.java Lines: 16
Jwc.java Lines: 32

# reading interactively until
Ctrl-D
$ javac Jwc.java && java Jwc
manual input
some more
some more  (user presses Ctrl-D)
Lines: 3
```

Main blocking for input - cont.

```
  static long readStdIn() {
    return new BufferedReader(new InputStreamReader(System.in))
        .lines()
        .count();
  }

  static void countFile(String[] args) {
    for (String file : args) {
      try {
        long lines = Files.lines(Paths.get(file)).count();
        System.out.println(file + " Lines: " + lines);
      } catch (IOException e) {
        System.err.println(e.getMessage());
      }
    }
  }
} // end class
```

Main handing off to an object

```
import javafx.application.Application;

public class HelloFXMain {

  public static void main(String[] args) {
    Application.launch(HelloFX.class, args);
    // Only when the application “dies” we get back to main!
  }

}
```

The application - part I

```
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.GridPane;
import javafx.geometry.Pos;
import javafx.geometry.Insets;
import javafx.stage.Stage;
```

The application - part II

```
public class HelloFX extends Application {

  private GridPane grid = new GridPane();
  private Label greeting = new Label("Hello");

  @Override
  public void start(Stage primaryStage) {
    Button btn = new Button();
    btn.setText("Greet!");
    btn.setOnAction(new EventHandler<ActionEvent>() {

        @Override
        public void handle(ActionEvent event) {
          if (getParameters().getRaw().size() != 0) {
            greeting.setText("Hello " + getParameters().getRaw().get(0));
          }
        }
      });
```

The application - part II

```
public class HelloFX extends Application {

  private GridPane grid = new GridPane();
  private Label greeting = new Label("Hello");

  @Override
  public void start(Stage primaryStage) {
    Button btn = new Button();
    btn.setText("Greet!");
    btn.setOnAction(new EventHandler<ActionEvent>() {

        @Override
        public void handle(ActionEvent event) {
          if (getParameters().getRaw().size() != 0) {
            greeting.setText("Hello " + getParameters().getRaw().get(0));
          }
        }
      });
```

The application - part III

```
    grid.setAlignment(Pos.CENTER);
    grid.setHgap(10);
    grid.setVgap(10);
    grid.setPadding(new Insets(25, 25, 25, 25));
    grid.add(btn,0,0);
    grid.add(greeting, 0, 1);
    Scene scene = new Scene(grid, 300, 250);
    primaryStage.setTitle("Hello World!");
    primaryStage.setScene(scene);
    primaryStage.show();
  }
}
```

Further reading

- Oracle Java Tutorial - Lesson: A Closer Look at the "Hello World!" Application
- Oracle - Essentials, Part 1, Lesson 2: Building Applications
- Wikipedia - Entry point
- Oracle's tutorial section on Classes
- Wikipedia on Java classes

Source code

The example Java files from this presentation can be found here:

https://github.com/progund/classes/tree/master/main-method