

# Tinker Academy

AP Computer Science Prep (Java Programming)  
(Java Execution Flow)

# Java Execution Flow

Statements & Blocks are always within a method, constructor

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("Starting Car Race...");  
    }  
}
```

Program Code within a Method

```
public class Car {  
    private String id;  
    public Car(String id) {  
        this.id = id;  
    }  
}
```

Program Code in a Constructor

First statement to be executed by the JVM is the first statement in **main**

# Java Execution Flow

1  
2  
48

EXIT

3  
4  
8  
9  
15  
17  
18

```
01 package com.tinkeracademy;
02
03 import com.tinkeracademy.race.Car;
04 import com.tinkeracademy.race.Race;
05
06 public class Main {
07
08     public static void main(String[] args) {
09         System.out.println("Starting Car Race...");
10         startRace(10, 10);
11         System.out.println("...Car Race Ended");
12     }
13
14     public static void startRace(int numCars, int numLaps) {
15         int lapDistance = 10000;
16         Race race = new Race(numLaps, lapDistance);
17         for (int i = 0; i < numCars; i++) {
18             Car car = new Car("Car " + i);
19             race.addCar(car);
20         }
21         race.race();
22     }
23 }
24
```

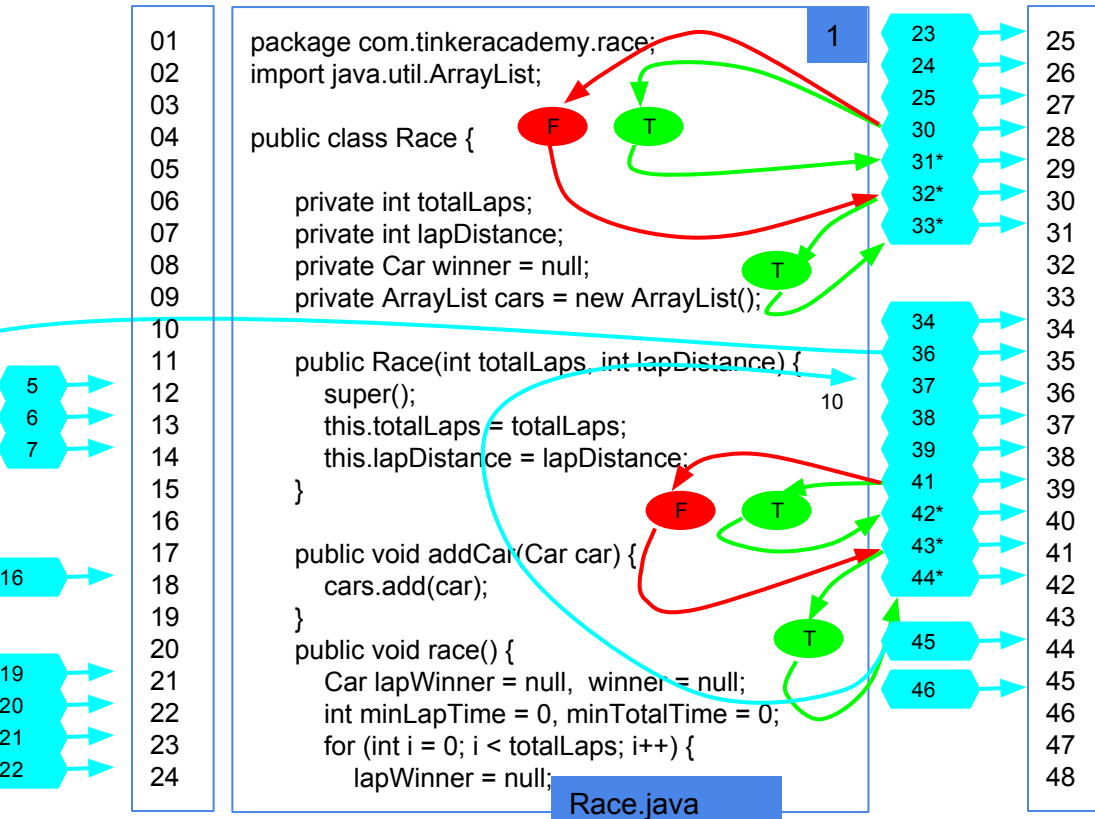
Main.java

1

25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48

2

# Java Execution Flow



2

```

for (int j = 0; j < cars.size(); j++) {
    Car car = (Car) cars.get(j);
    int lapTime = car.race(lapDistance);
    if (lapWinner == null) {
        lapWinner = car; minLapTime = lapTime;
    } else if (lapTime < minLapTime) {
        lapWinner = car; minLapTime = lapTime;
    }
}
System.out.println(lapWinner + " won lap " + i);
}
for (int i = 0; i < cars.size(); i++) {
    Car car = (Car) cars.get(i);
    int totalTime = car.getTotalTime();
    if (winner == null) {
        winner = car; minTotalTime = totalTime;
    } else if (totalTime < minTotalTime) {
        winner = car; minTotalTime = totalTime;
    }
}
System.out.println(winner + " wins race in " +
    minTotalTime + " secs!");
}
    
```

# Java Execution Flow

```
01 package com.tinkeracademy.race;
02
03 import com.tinkeracademy.Vehicle;
04
05 public final class Car extends Vehicle {
06
07     private String id;
08     private int totalTime;
09
10     public Car(String id) {
11         super();
12         if (id == null) {
13             throw new IllegalArgumentException("null id");
14         }
15         this.id = id;
16     }
17
18     public int race(int lapDistance) {
19         int lapSpeed = 100 + (int) (Math.random() * 100);
20         int lapTime = (lapDistance / lapSpeed);
21         totalTime += lapTime;
22         return lapTime;
23     }
24 }
```

1

40

35, 47

```
25
26     return totalTime;
27 }
28
29     public String toString() {
30         return id;
31     }
32
33 }
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
```

2

# Java Execution Flow

11 →

```
01 package com.tinkeracademy;  
02  
03 public abstract class Vehicle implements IVehicle {  
04  
05     public Vehicle() {  
06         super();  
07     }  
08  
09     abstract public int getTotalTime();  
10  
11 }  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24
```

Vehicle.java

1

```
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48
```

2

# Java Execution Flow

```
01 package com.tinkeracademy;
02
03 public interface IVehicle {
04     /**
05      * Returns the total time traveled by vehicle
06      * in seconds since it last started
07      */
08     int getTotalTime();
09 }
```

IVehicle.java

1

```
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
```

2