Tinker Academy

AP Computer Science Prep (Java Programming)

(Java Execution Flow)

Spring 2015

Java Execution Flow

Statements & Blocks are always within a method, constructor

```
public class Main {
    public static void main(String[] args) {
1       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

```
01
02
03
04
05
06
07
08
09
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
```

```
package com.tinkeracademy;

import com.tinkeracademy.race.Car;
import com.tinkeracademy.race.Race;

public class Main {

    public static void main(String[] args) {
        System.out.println("Starting Car Race...");
        startRace(10, 10);
                System.out.println("...Car Race Ended");
    }

    public static void startRace(int numCars, int numLaps) {
        int lapDistance = 10000;
        Race race = new Race(numLaps, lapDistance);
        for (int i = 0; i < numCars; i++) {
            Car car = new Car("Car " + i);
            race.addCar(car);
        }
        race.race();
    }
}
```

Main.java

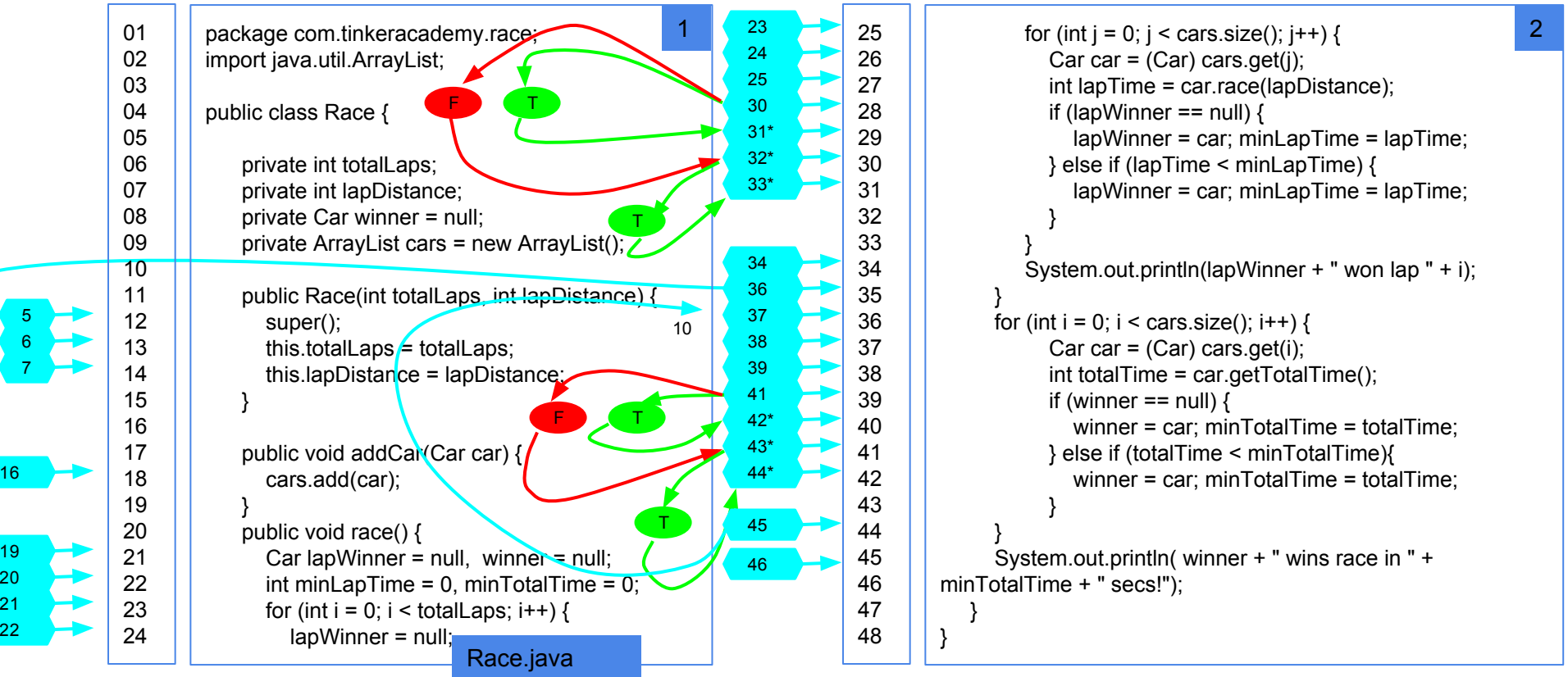
2

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

1 → 2 → 48 → EXIT

3 → 4 → 8 → 9 → 15 → 17 → 18

Java Execution Flow



Java Execution Flow

```
01
02
03
04
05
06
07
08
09
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
```

1

```
package com.tinkeracademy.race;

import com.tinkeracademy.Vehicle;

public final class Car extends Vehicle {

    private String id;
    private int totalTime;

    public Car(String id) {
        super();
        if (id == null) {
            throw new IllegalArgumentException("null id");
        }
        this.id = id;
    }

    public int race(int lapDistance) {
        int lapSpeed = 100 + (int) (Math.random() * 100);
        int lapTime = (lapDistance / lapSpeed);
        totalTime += lapTime;
        return lapTime;
    }
```

Car.java

10
12
13*
14
26
27
28
29

40

35, 47

```
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

```
    public int getTotalTime() {
        return totalTime;
    }

    public String toString() {
        return id;
    }

}
```

Java Execution Flow

11

```
01
02
03
04
05
06
07
08
09
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
```

1

```
package com.tinkeracademy;

public abstract class Vehicle implements IVehicle {

    public Vehicle() {
        super();
    }

    abstract public int getTotalTime();

}
```

Vehicle.java

```
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

1

```
01
02
03
04
05
06
07
08
09
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
```

```
package com.tinkeracademy;

public interface IVehicle {
    /**
     * Returns the total time traveled by vehicle
     * in seconds since it last started
     */
    int getTotalTime();
}
```

IVehicle.java

2

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