COM6516 Object Oriented Programming and Software Design

The contents of this module has been developed by Adam Funk, Kirill Bogdanov, Mark Stevenson, Richard Clayton and Heidi Christensen

Practical 1

Introduction to Java

- Running a simple Java program
- Type promotion
- I/O from a file
- Enumerations
- Comments
- Write, compile & run

Simple Java program

```
$ javac HelloWorld.java
HelloWorld.java:26: error: cannot find symbol
System.out.println( helloworldString );
                    ^
  symbol:   variable helloworldString
  location: class HelloWorld
1 error
```

This means the compiler does not recognize the identifier; common causes:

- typographical/spelling mistakes
- forgetting to declare variables (methods, etc.)

Simple Java program

Here's the error:

```
String helloString = "Hello";
String worldString = "World!";
...
System.out.println( helloworldString );
```

Correction: insert this before the println:

```
String helloWorldString = helloString + " " + worldString;
```

or just change the println argument:

```
System.out.println(helloString + " " + worldString);
```

Type promotion

`TypeCast`:

- If both arguments are integer, the result is integer

  (e.g.) `3/2 → 1`

- If both arguments are numeric and at least one is double, the result is double

  (e.g.) `3.0/2 → 1.5; 3/2.0 → 1.5; 3.0/2.0 → 1.5;`

- If + involves a String and a number, the number is converted to a String for concatenation → result is String

  (e.g.) `“foo”+3 → “foo3”`

- Note that `+` works from left to right — see `QuadraticSolver` for an example where mixing strings and numbers can go wrong

Type promotion

`QuadraticSolver`:

- Sample code to test the solution:

```
System.out.println("Testing with double:");
System.out.println("quadratic x1 = " +
    (aFloat * x1 * x1 + bFloat * x1 + cFloat));
System.out.println("quadratic x2 = " +
    (aFloat * x2 * x2 + bFloat * x2 + cFloat));
```

- Results:

```
Testing with double:
quadratic x1 = 1.1102230246251565E-16
quadratic x2 = 0.0
```

+ is **overloaded** (it does more than one thing) and works from left to right

Type promotion

`QuadraticSolver`:

- Suppose we remove ( ) around the arithmetic:

```
System.out.println("Testing with double:");
System.out.println("quadratic x1 = " +
    aFloat * x1 * x1 + bFloat * x1 + cFloat);
System.out.println("quadratic x2 = " +
    aFloat * x2 * x2 + bFloat * x2 + cFloat);
```

- Results:

```
Testing with double:
quadratic x1 = 0.17157287525380985-1.17157287525380971.0
quadratic x2 = 5.82842712474619-6.82842712474619 1.0
```

It prints *ax*^2, *bx*, and *c* without spaces!

I/O from a file

Using the `EasyReader` class in the sheffield package

- Look closely at the example (`KeyboardInput.java`):

```
// step 1, create a new EasyReader object to model the keyboard
EasyReader keyboard = new EasyReader();

// step 2, prompt the user to input values for a, b, and c and
// store these as doubles
double a = keyboard.readDouble("Input a value for a: ");
```

I/O from a file

Look at `EasyReader` code — either html documentation or directly in the code

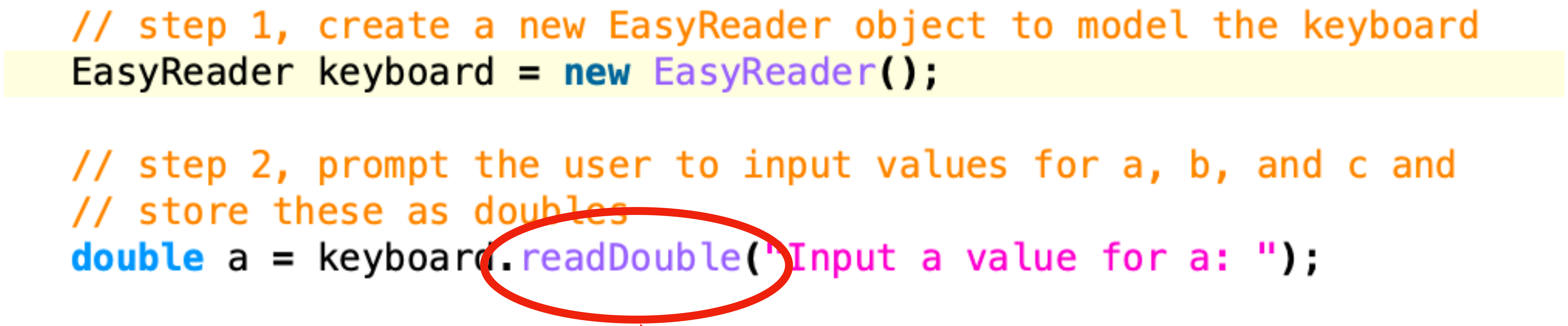
```
/**
* Read an integer from the input stream
* @return an integer value
*/
public int readInt() {
    int x = 0;
    try {
        x=(new Integer(readString())).intValue();
        }
    catch (Exception e) {
        error("invalid integer number");
        }
    return x;
    }

/**
* Read an integer from the input stream, with a prompt
* @return an integer value
*/
public int readInt(String s) {
    prompt(s);
    return readInt();
    }
```

I/O from a file

Using the `EasyReader` class in the sheffield package

- Look closely at the example (`KeyboardInput.java`):



```
// step 1, create a new EasyReader object to model the keyboard
EasyReader keyboard = new EasyReader();

// step 2, prompt the user to input values for a, b, and c and
// store these as doubles
double a = keyboard.readDouble("Input a value for a: ");
```

replace with `readInt`

I/O from a file

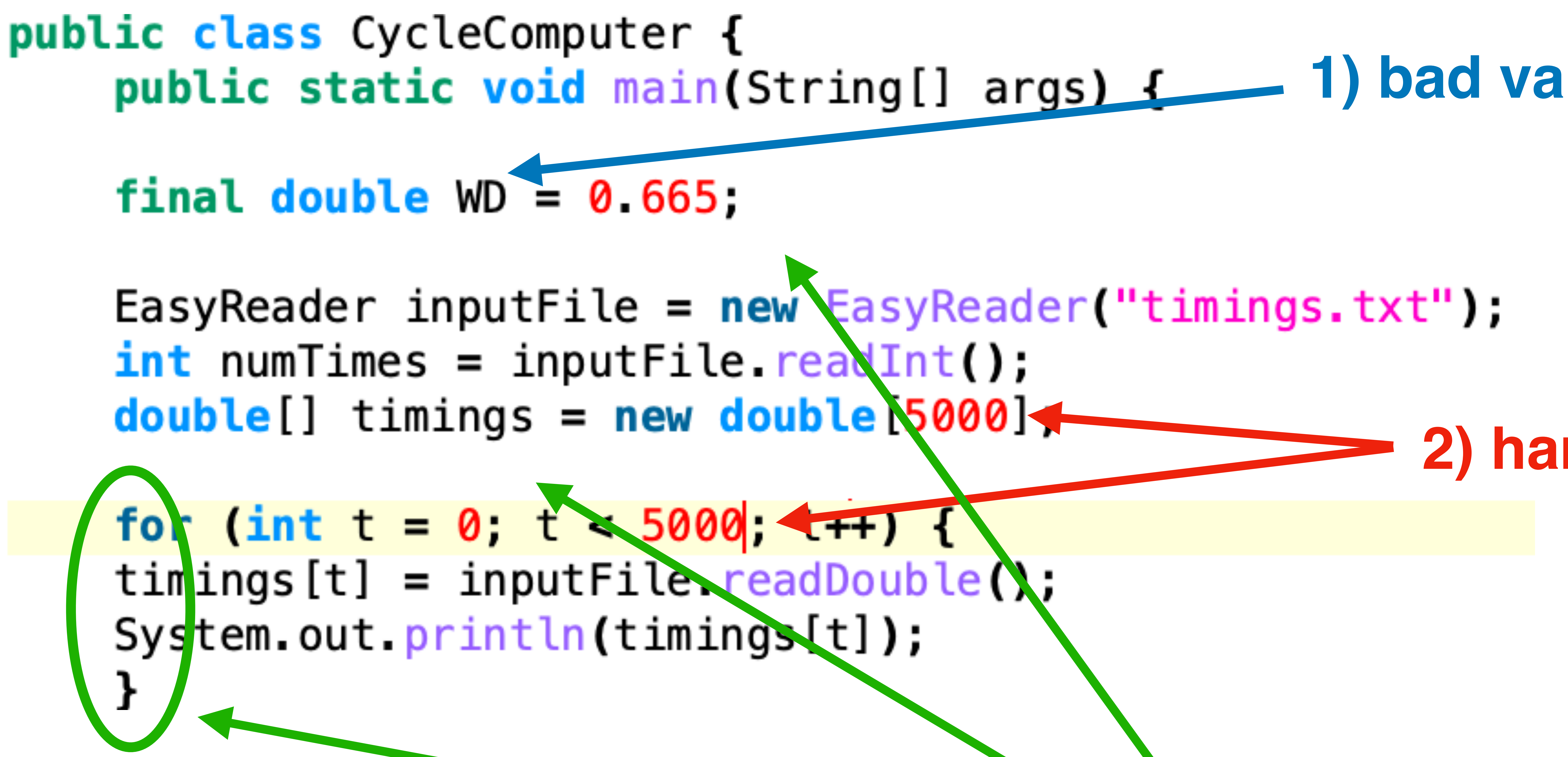
```java
public class CycleComputer {
    public static void main(String[] args) {

        // set wheel diameter in m, 665 mm is correct for 26 inch wheel plus tyre
        // this variable is final because we do not need to change it
        final double WHEEL_DIAMETER = 0.665;

        EasyReader inputFile = new EasyReader("timings.txt");
        int numTimes = inputFile.readInt();
        double[] timings = new double[numTimes];

        // read in all the timings, each time interval in seconds
        for (int t = 0; t < numTimes; t++) {
            timings[t] = inputFile.readDouble();
            System.out.println(timings[t]);
        }
```

I/O from a file



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

    final double WD = 0.665;

    EasyReader inputFile = new EasyReader("timings.txt");
    int numTimes = inputFile.readInt();
    double[] timings = new double[5000];

    for (int t = 0; t < 5000; t++) {
    timings[t] = inputFile.readDouble();
    System.out.println(timings[t]);
    }
```

1) bad variable names

2) hard-coded numbers

3) missing comments and missing indentation

Bad practice examples

Enumerations

- An enumeration is a class that lists all the objects that may be created:
  `public enum Color { RED, GREEN, BLUE };`
- You can then declare a variable of type `COL`
  ```
  Color c = Color.RED;
  Color d = Color.GREEN;
  ```
- Comparison between variables of enumerated type is easy:
  `if (c == Color.GREEN)…`
  or
  `if (c == d)…`
- Since the only instances of enumerated type are listed in the `enum` declaration, comparison by reference works well

Enumerations

- You can use a `switch` statement:

```
switch(c) {
  case RED: // something red
    break;
  case GREEN: // something green
    break;
  // …
}
```

- You can iterate through all values of an enumerated type:

```
for(Color s:Color.values())…
```

Enumerations: loading data

Imagine you have a text file and would like to load data from it

```
Color newC = Color.valueOf(s)
```

will take string `s` and return an instance of enumerated type `Color`

- Input: `RED GREEN MEERKAT`
- If there is no corresponding value, an `IllegalArgumentException` is thrown

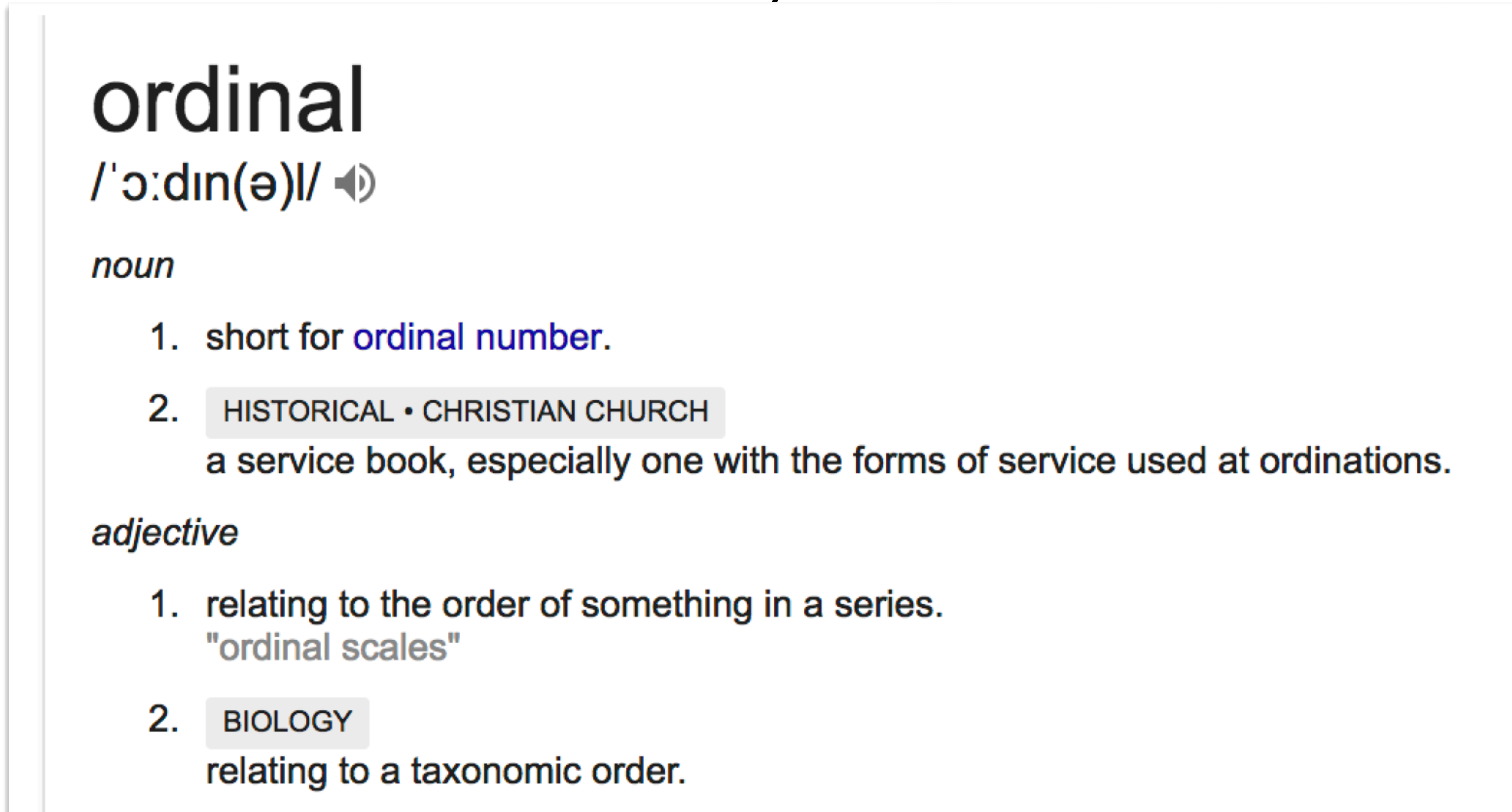
Enumerations

For more information and examples:

https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html

- days of the week
- planets with their mass and radius (data attached to each item)

Enumerations come with the built-in methods `toString()` (obvious value) and `ordinal()` (less useful in most cases)



ordinal

/ˈɔːdɪn(ə)l/

noun

1. short for ordinal number.
2. HISTORICAL • CHRISTIAN CHURCH
   a service book, especially one with the forms of service used at ordinations.

adjective

1. relating to the order of something in a series.
   "ordinal scales"
2. BIOLOGY
   relating to a taxonomic order.

Enumerations

```
public class Test {
  public enum Color {RED, BLUE, GREEN};
  public static void main(String[] args) {
    for (Color c : Color.values()) {
      System.out.println(c.toString() + " " + c.ordinal());
    }
  }
}
```

Output:

```
RED 0
BLUE 1
GREEN 2
```

Enumerations

```
public class Test {
  public enum Color {RED, BLUE, GREEN};
  public static void main(String[] args) {
    if (Color.RED.ordinal() <
        Color.BLUE.ordinal()) {
          System.out.println("Red < blue!");
    }
  }
}
```

Output:

```
Red < blue!
```

Enumerations

```
public class Test {
  public enum Day {MON, TUE, WED, THU, FRI, SAT, SUN};

  public static void test(Day day) {
    if (day.ordinal() < Day.SAT.ordinal()) {
      System.out.println("Get back to work!");
    }
  }
}
```

This wouldn’t work the same way if we started with `SUN`

Enumerations

```
public class Test {
  public enum Color {RED, BLUE, GREEN};
  public Color currentColor;
  // …
}
```

In other classes, refer to the enum as a nested class:

```
public class AnotherClass {
  public void setLocalColor(Test.Color c) {
    // …
  }
}
```

Enumerations

An enum can also be a class in its own file, e.g., `AnimalType.java` (treat as a normal class in other code):

```
public class AnimalType {
  GOAT, DOG, CAT, FISH
}
```

Comments

Format of comments in Java is similar to C, C++.

Single line - Begins with // and continues to the end of the line, (e.g.)

```
double distance = 20;  // distance in miles
double time;
```

Multi line - Begins with /* and ends with */, (e.g.)

```
/*
This block converts miles to km
*/
```

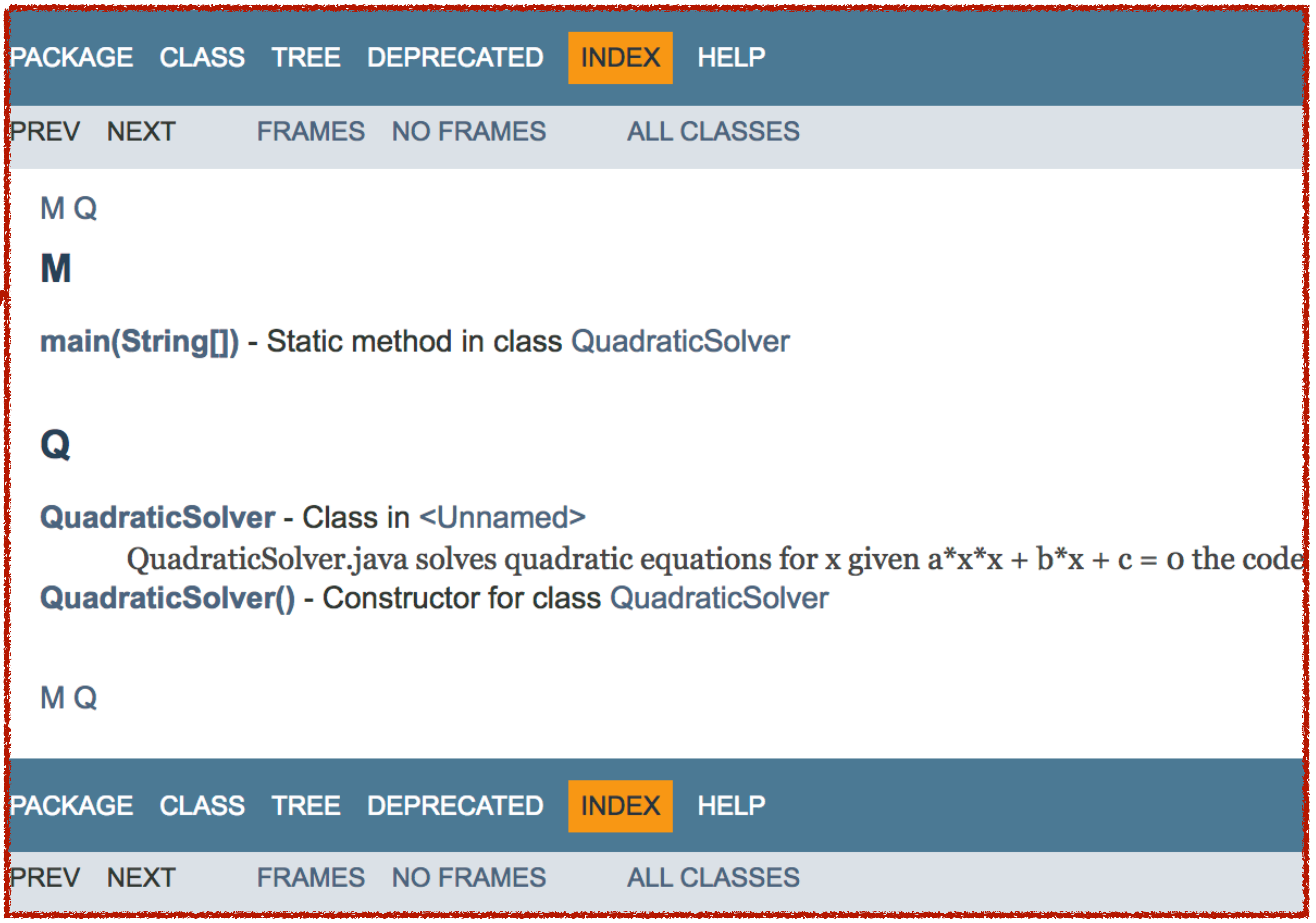
Note that multi line comments cannot be nested in Java.

Comments

Javadoc Special comments for documentation with HTML tags and a simple syntax understood by *javadoc*. Begins with /** and ends with */

```
/**
* Convert miles to kilometers.
* @param distance Distance in miles
*/
```

```
>javadoc QuadraticSolver.java
```


PACKAGE CLASS TREE DEPRECATED INDEX HELP
PREV NEXT FRAMES NO FRAMES ALL CLASSES
M Q

M

main(String[]) - Static method in class QuadraticSolver

Q

QuadraticSolver - Class in <Unnamed>

QuadraticSolver.java solves quadratic equations for x given a*x*x + b*x + c = 0 the code

QuadraticSolver() - Constructor for class QuadraticSolver

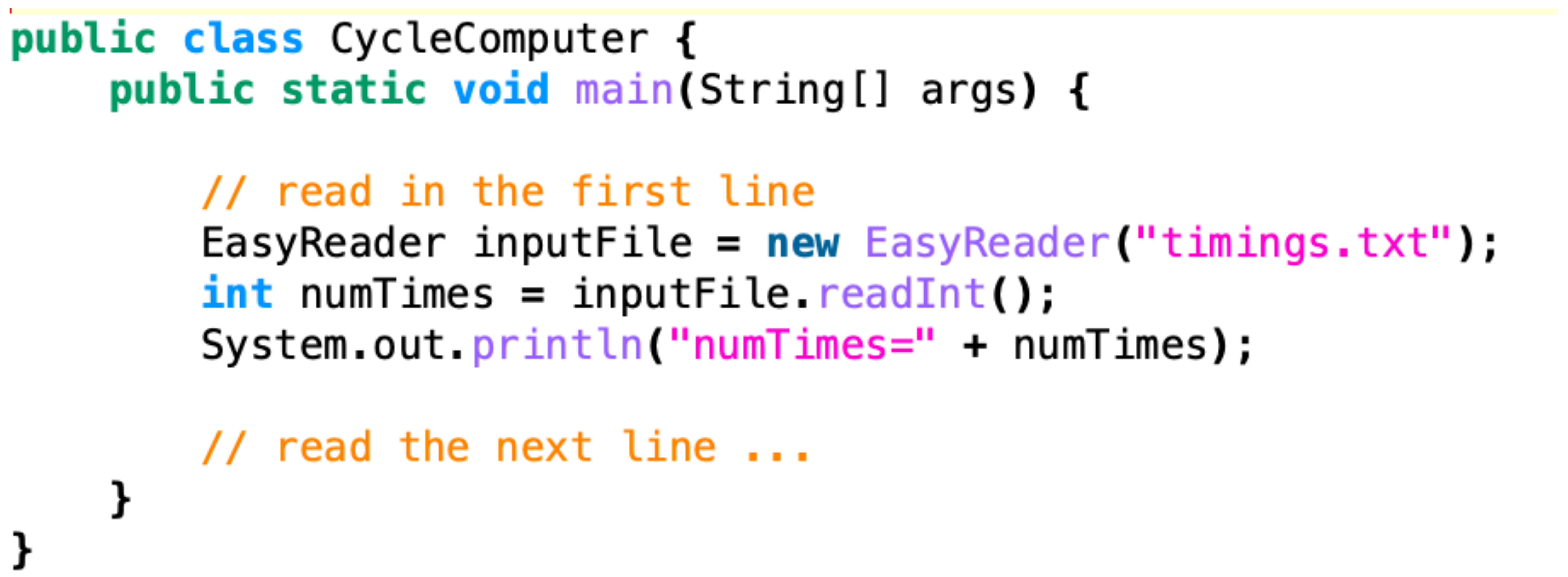
M Q
PACKAGE CLASS TREE DEPRECATED INDEX HELP
PREV NEXT FRAMES NO FRAMES ALL CLASSES

Write, compile & run

```java
public class CycleComputer {
    public static void main(String[] args) {
        System.out.println("Hurrah, I've compiled!");
    }
}
```

compile & run

Write, compile & run



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

        // read in the first line
        EasyReader inputFile = new EasyReader("timings.txt");
        int numTimes = inputFile.readInt();
        System.out.println("numTimes=" + numTimes);

        // read the next line ...
    }
}
```

compile & run

Write, compile & run

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

        // set wheel diameter in m, 665 mm is correct for 26 inch wheel plus tyre
        // this variable is final because we do not need to change it
        final double WHEEL_DIAMETER = 0.665;

        EasyReader inputFile = new EasyReader("timings.txt");
        int numTimes = inputFile.readInt();
        double[] timings = new double[numTimes];

        // read in all the timings, each time interval in seconds
        for (int t = 0; t < numTimes; t++) {
            timings[t] = inputFile.readDouble();
            System.out.println(timings[t]);
        }
```

compile & run