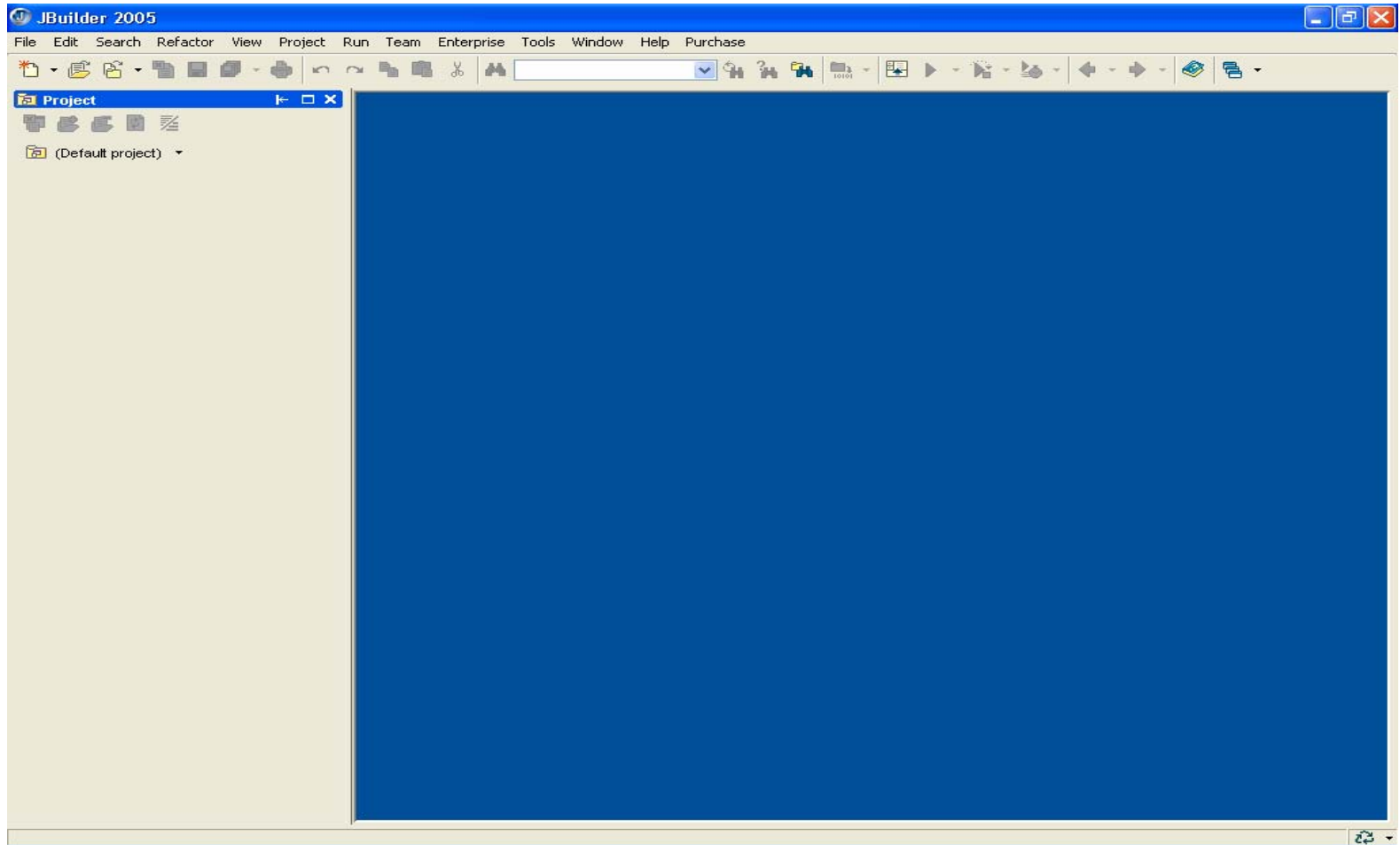


Java

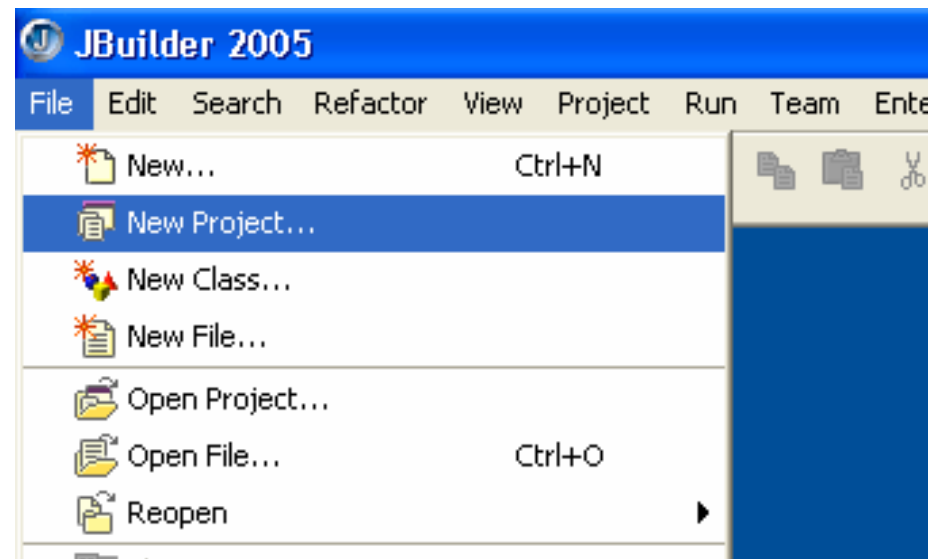
Dr. Omar Hamed
omarh@arabou.org.sa

Using JBuilder 2005



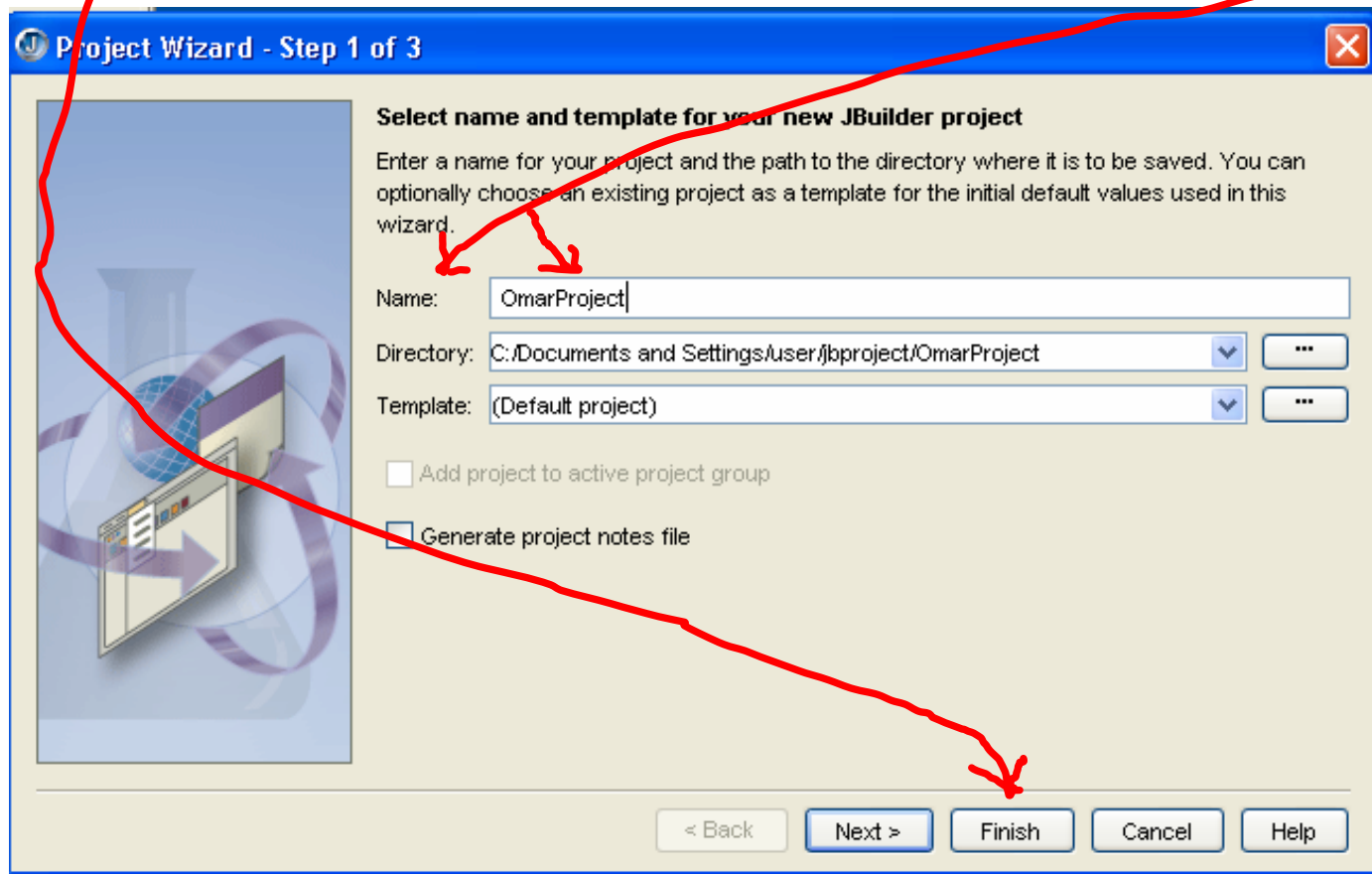
JBuilder

- Make sure to Close all projects before starting your new project. This can be done using
- File->close projects->all->ok
- Press File → new project (see picture)



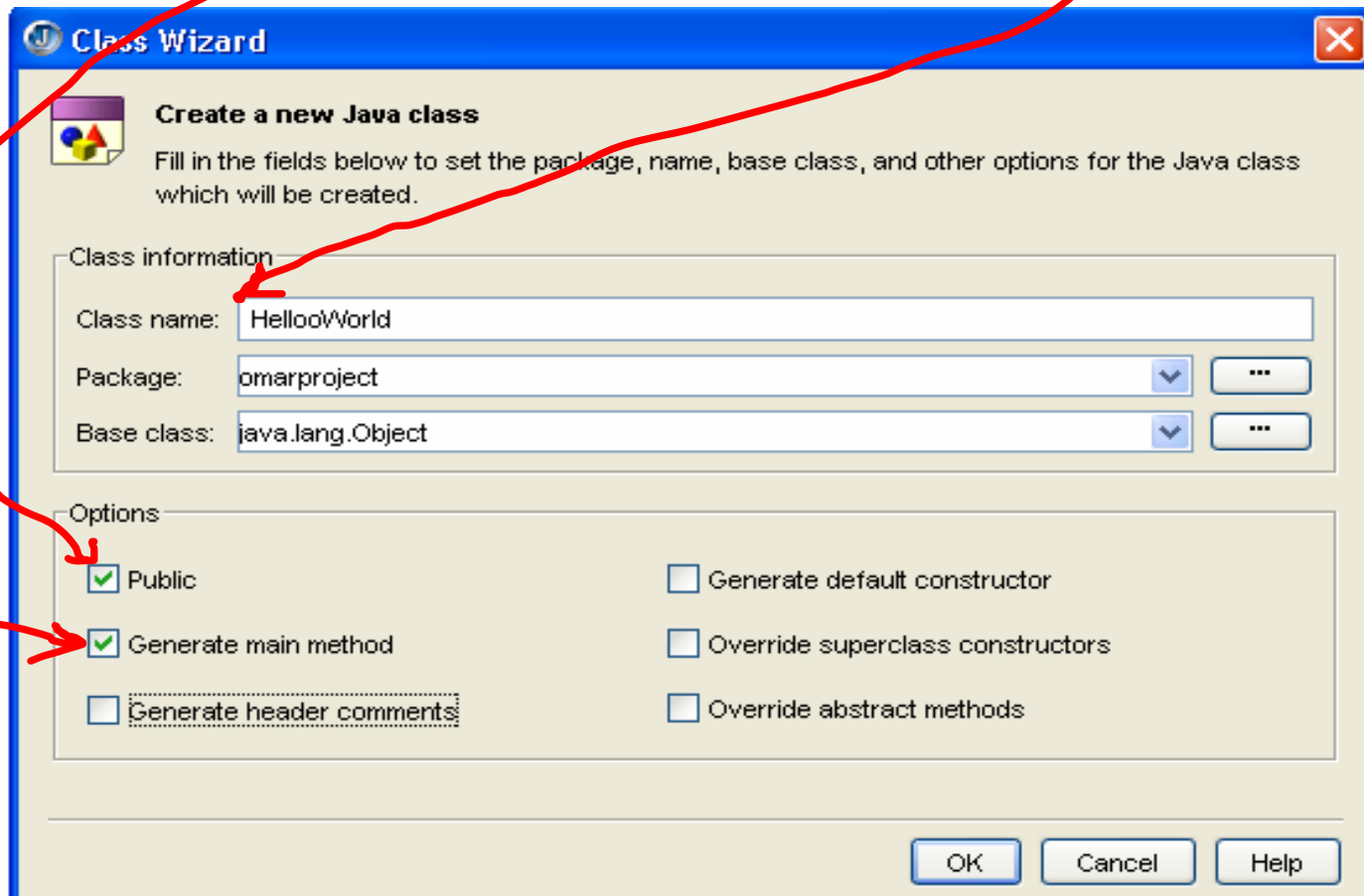
JBuilder

- You will be provided with a window to specify the name of the project as shown (I called the new project **OmarProject**) then press finish.

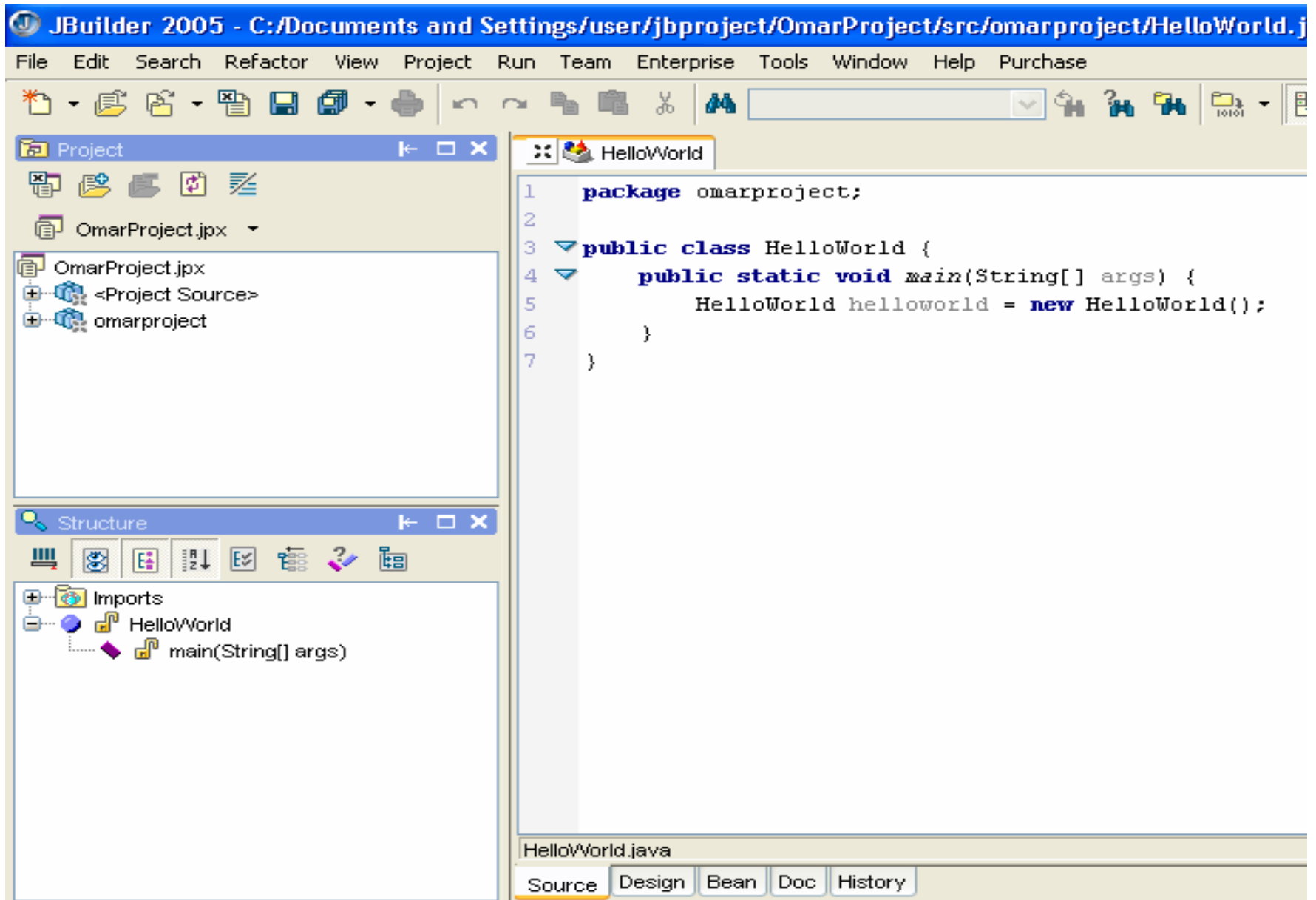


JBuilder

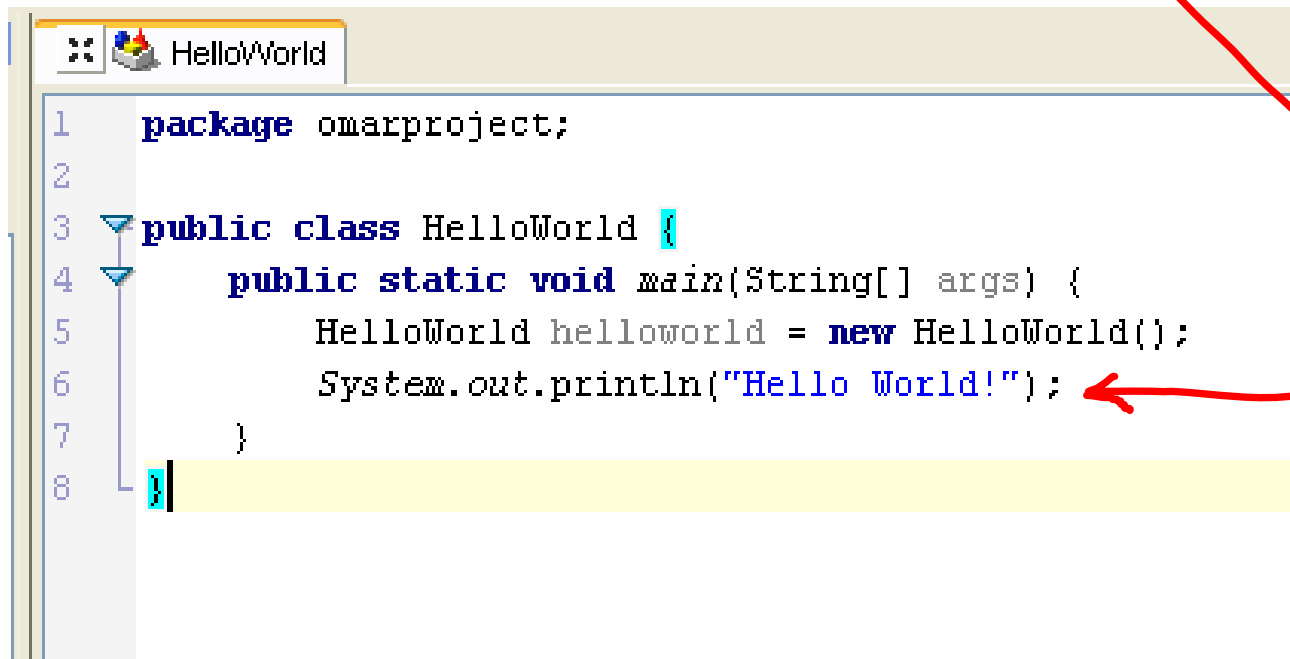
- Create a new class using File→new Class
- A window will pop up to name the class.
 - Mark public & Generate main method (see below)



JBuilder will Build the class as shown



Add the statement shown

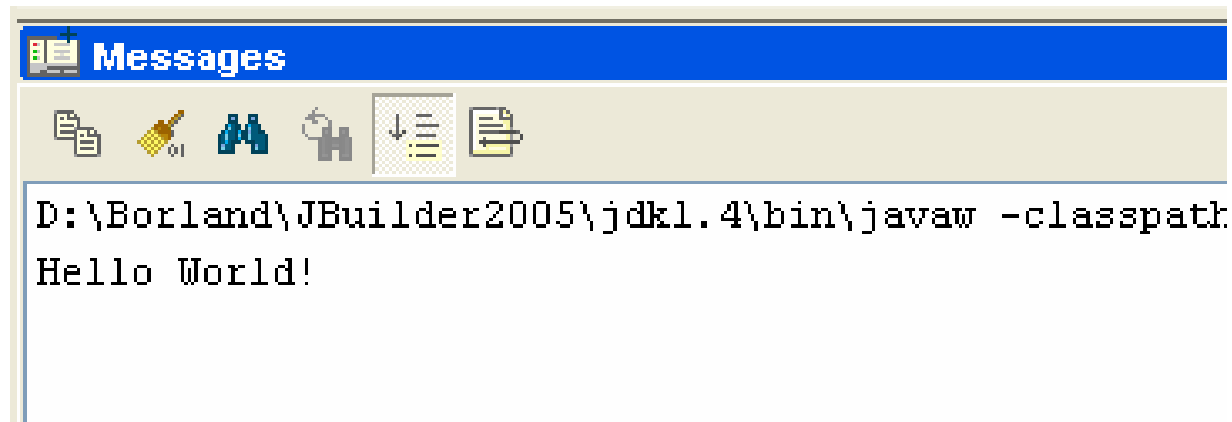


```
1 package omarproject;
2
3 public class HelloWorld {
4     public static void main(String[] args) {
5         HelloWorld helloworld = new HelloWorld();
6         System.out.println("Hello World!");
7     }
8 }
```

Press the run icon



to get the required results
that is printing Hello World



Examining Hello World

The initial `class` statement
may be thought of as defining the program name,
in this case HelloWorld.

It's advisable
to give the source code file
the same name as the main class in the file
plus the .java extension.

The HelloWorld class
contains
one *method*,
the **main** method.

The main method is
where an application **begins executing**.

The method is declared
public
meaning that the method can be called from anywhere.

It is declared
static
meaning that **all** instances of this class **share** this one method.

It is declared

void

which means that this method
does not return a value.

We *pass* any command line arguments to the method in an array of Strings
called *args*.

In this simple program there aren't any command line arguments though.

Finally when the main method is called
it does exactly one thing:

`print "Hello World"`

This is accomplished by
calling the

println() method

of the **static out field**

Belonging

to the **System class**

Braces and Blocks

In Java a source code file
is broken up into **parts**
separated by opening and closing **braces**,
i.e. the

{

And

}

characters.

Everything between { and } is a **block**
and exists more or less
independently of everything outside of the braces.

Blocks

can be hierarchical.

One block can contain
one or more subsidiary blocks.

In our example we have
one outer block
that defines the HelloWorld class.
Within the HelloWorld block
we have
a method block
called "main".

Comments

Comments can appear anywhere in a source file.

Everything between

/*

And

***/**

is ignored by the compiler
and everything on a line
after two consecutive slashes

//

is also thrown away.

e.g

// This is the Hello World program in Java

The general form of **IF**

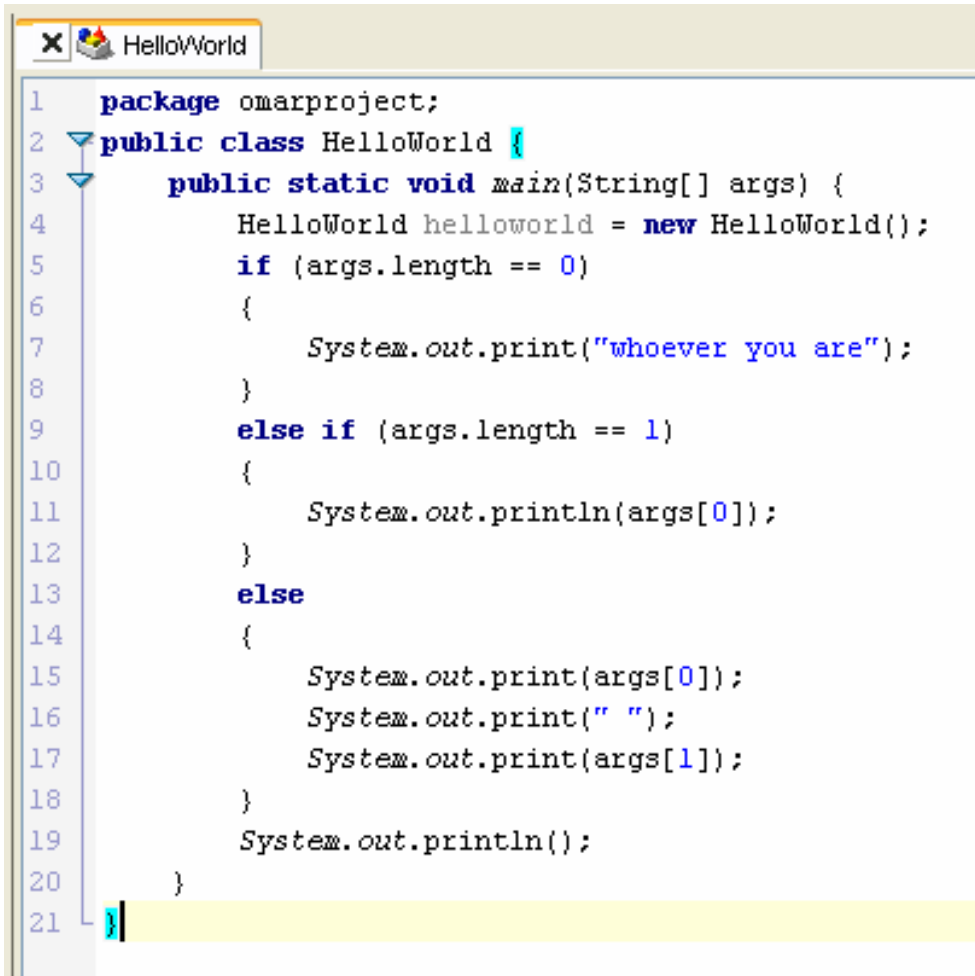
MR IF

```
if (args.length > 0) {  
    System.out.println(args[0]);  
}
```

MR IF & MR ELSE

```
if (args.length > 0) {  
    System.out.println(args[0]);  
}  
else {  
    System.out.println("whoever you are");  
}
```

Mr If – ELSE IF - ELSE



```
1 package omarproject;
2 public class HelloWorld {
3     public static void main(String[] args) {
4         HelloWorld helloworld = new HelloWorld();
5         if (args.length == 0)
6         {
7             System.out.print("whoever you are");
8         }
9         else if (args.length == 1)
10        {
11            System.out.println(args[0]);
12        }
13        else
14        {
15            System.out.print(args[0]);
16            System.out.print(" ");
17            System.out.print(args[1]);
18        }
19        System.out.println();
20    }
21 }
```

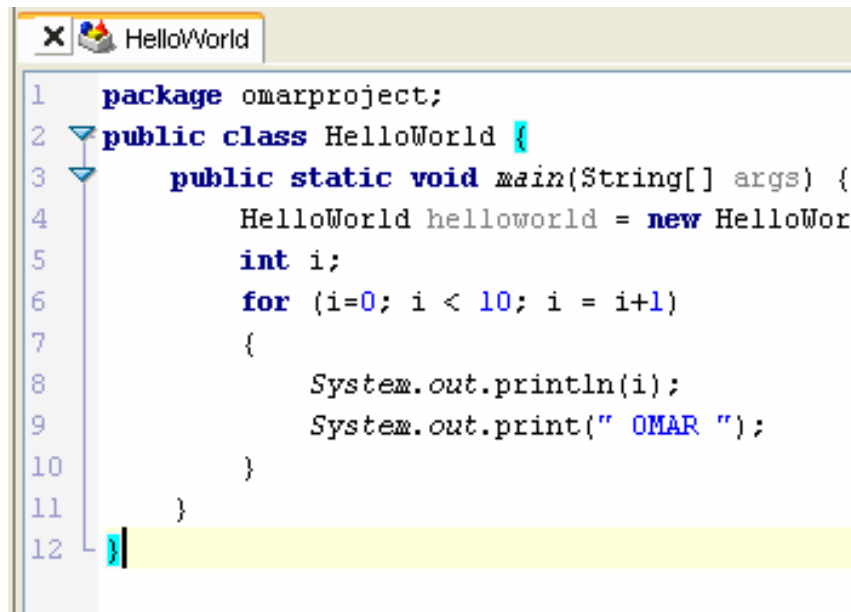
What is the output if the data was

ARAB OPEN UNIVERSIRTY

What is the output if the data was

“ARAB OPEN UNIVERSIRTY”

MR FOR

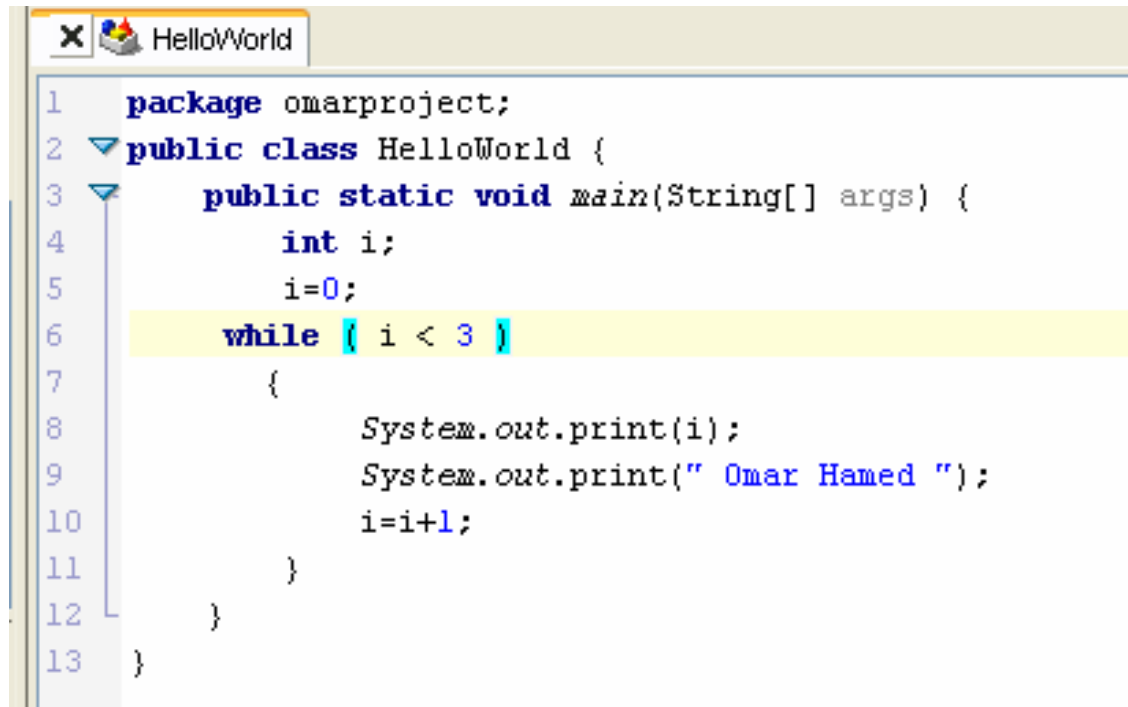


The screenshot shows a Java IDE window titled "HelloWorld". The code is as follows:

```
1 package omarproject;
2 public class HelloWorld {
3     public static void main(String[] args) {
4         HelloWorld helloworld = new HelloWor
5         int i;
6         for (i=0; i < 10; i = i+1)
7         {
8             System.out.println(i);
9             System.out.print(" OMAR ");
10        }
11    }
12 }
```

The code is a Java program that prints the numbers 0 through 9 and the name "OMAR" on the same line. The IDE has a light blue background with a vertical line on the left side of the code editor.

Mr while



```
1 package omarproject;
2 public class HelloWorld {
3     public static void main(String[] args) {
4         int i;
5         i=0;
6         while ( i < 3 )
7         {
8             System.out.print(i);
9             System.out.print(" Omar Hamed ");
10            i=i+1;
11        }
12    }
13 }
```

We can use

i++ In place of **i = i + 1.**

&

i-- for **i = i - 1.**

Adding and subtracting one from a number are such common operations

that these special

increment and ***decrement***

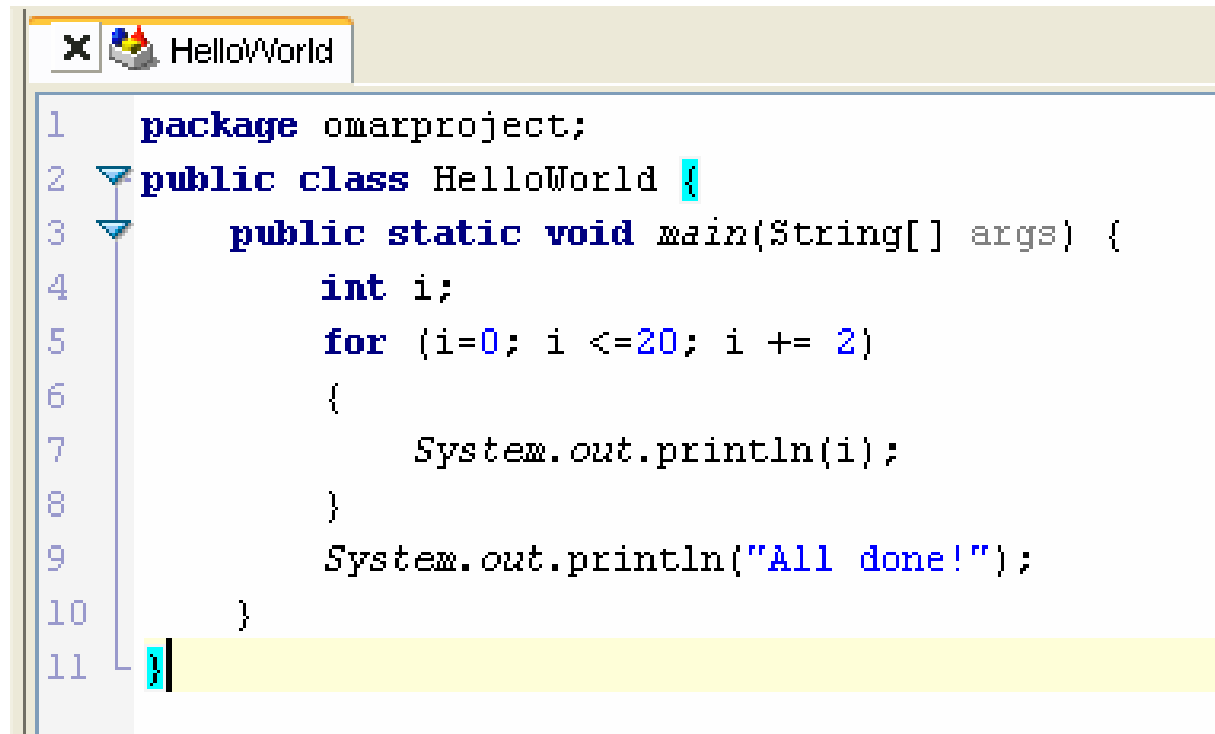
operators have been added to the language.

i=i+14 can be written as **i +=14**

&

i=i-3 can be written as **i -=3**

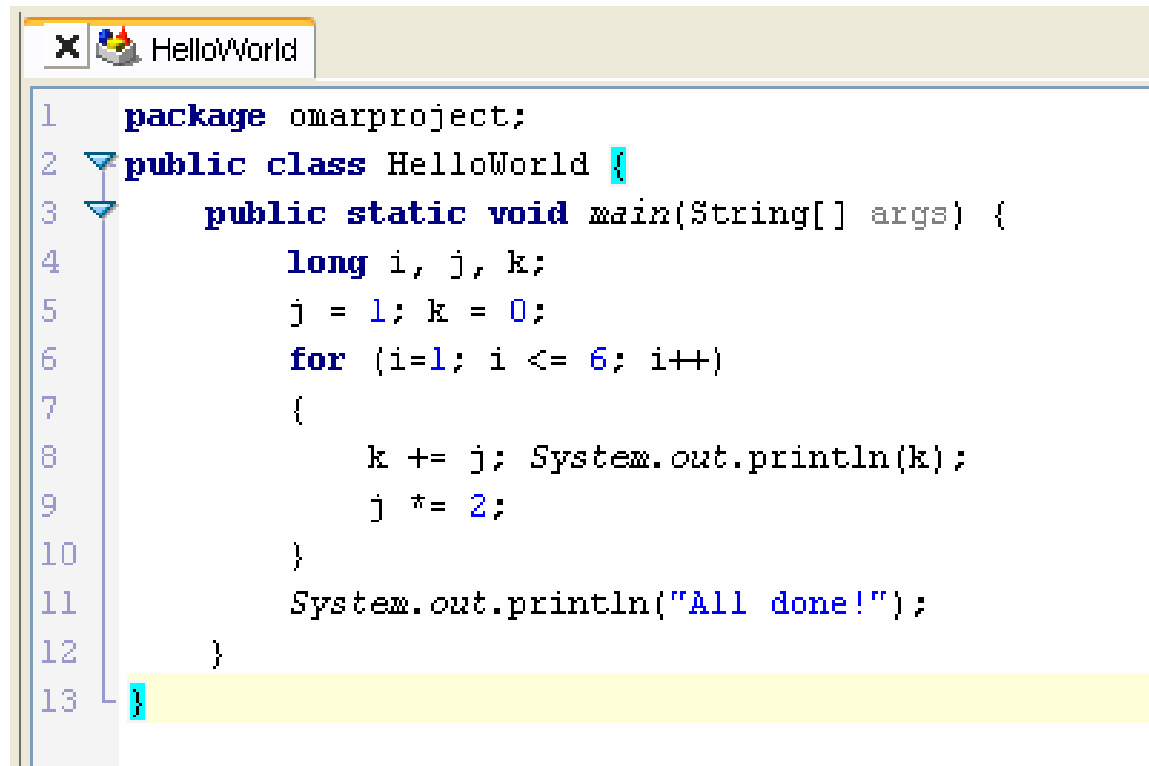
What are the results?

A screenshot of a Java IDE window titled 'HelloWorld'. The code is as follows:

```
1 package omarproject;
2 public class HelloWorld {
3     public static void main(String[] args) {
4         int i;
5         for (i=0; i <=20; i += 2)
6         {
7             System.out.println(i);
8         }
9         System.out.println("All done!");
10    }
11 }
```

The code is written in a monospaced font with syntax highlighting. The package name is 'omarproject'. The class is 'HelloWorld'. The main method iterates from 0 to 20 in increments of 2, printing each value. After the loop, it prints 'All done!'. The code is enclosed in a package declaration and a class declaration. The IDE window has a standard title bar with a close button and a file icon.

What are the results?

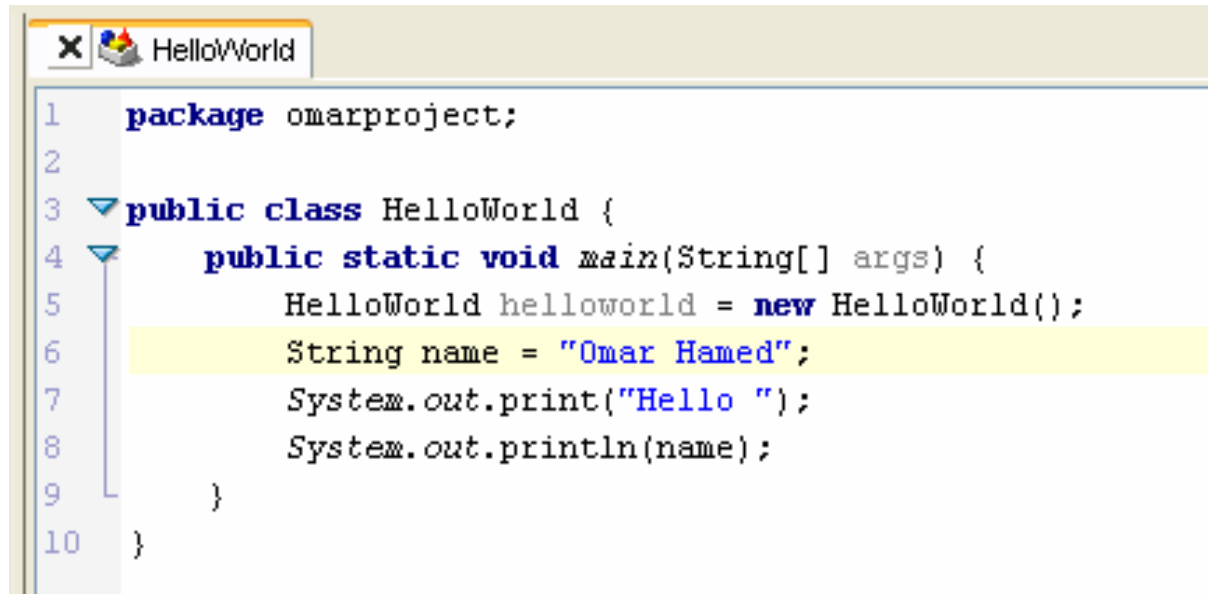
A screenshot of a Java IDE window titled 'HelloWorld'. The code is as follows:

```
1 package omarproject;
2 public class HelloWorld {
3     public static void main(String[] args) {
4         long i, j, k;
5         j = 1; k = 0;
6         for (i=1; i <= 6; i++)
7         {
8             k += j; System.out.println(k);
9             j *= 2;
10        }
11        System.out.println("All done!");
12    }
13 }
```

The code is a Java program in the 'omarproject' package. It defines a 'HelloWorld' class with a 'main' method. Inside the 'main' method, it declares three 'long' variables: 'i', 'j', and 'k'. 'j' is initialized to 1 and 'k' to 0. A 'for' loop runs from 'i=1' to 'i=6'. In each iteration, 'k' is incremented by 'j' and the result is printed. Then 'j' is doubled. After the loop, a final message 'All done!' is printed. The closing brace of the 'main' method and the class are highlighted in yellow.

Data and Variables

Consider the following generalization of the HelloWorld program:

A screenshot of a Java IDE window titled 'HelloWorld'. The code is as follows:

```
1 package omarproject;
2
3 public class HelloWorld {
4     public static void main(String[] args) {
5         HelloWorld helloworld = new HelloWorld();
6         String name = "Omar Hamed";
7         System.out.print("Hello ");
8         System.out.println(name);
9     }
10 }
```

The line `String name = "Omar Hamed";` is highlighted in yellow.

Here we allow Java to say hello to a specific individual. We do this by creating a

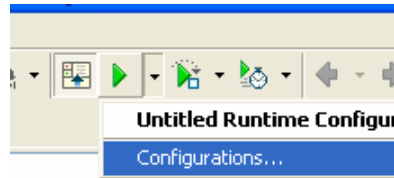
String variable

called "name" and storing the value "Omar Hamed" in it.

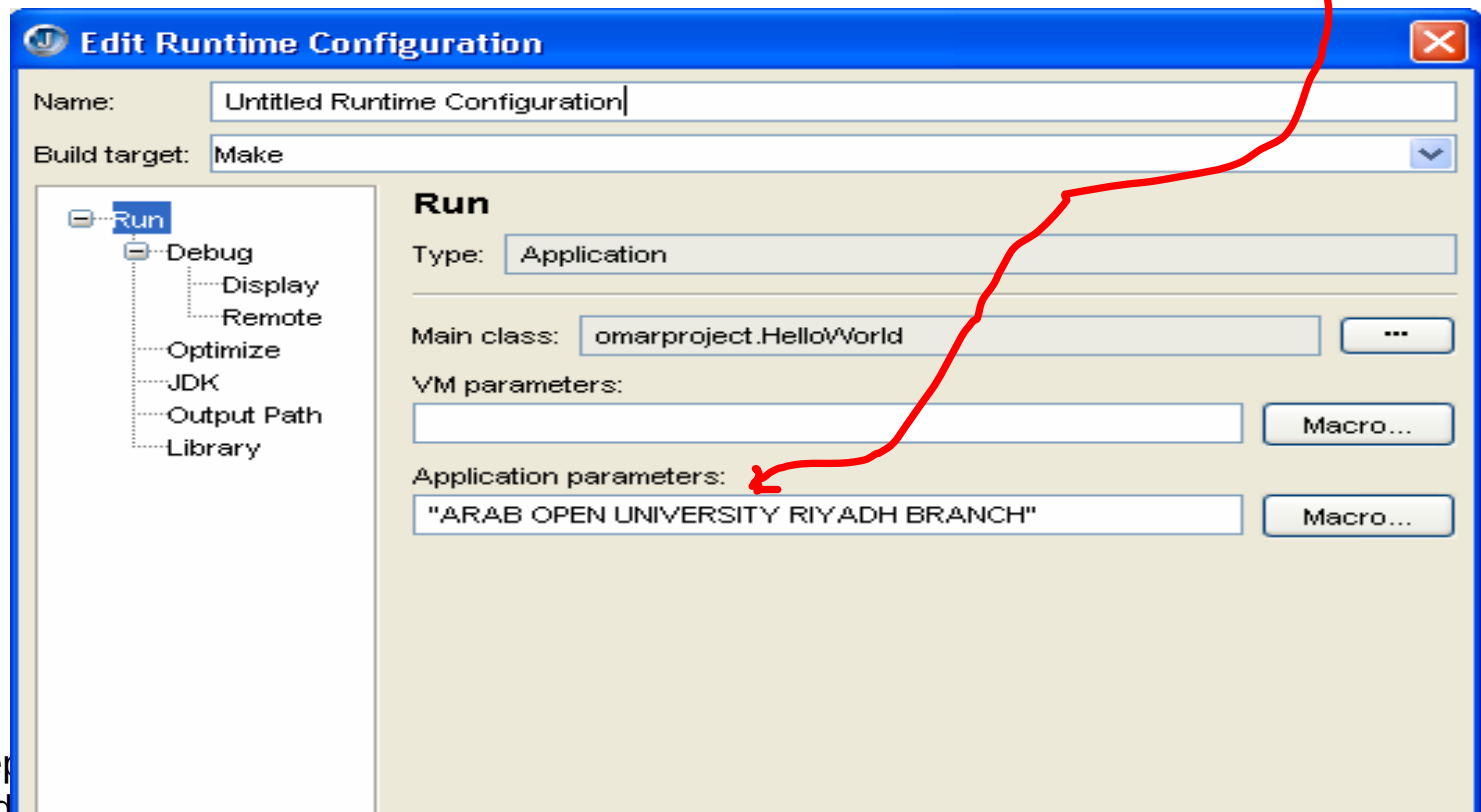
Data and Variables

What if we want to change the name at *runtime* rather than at compile time.

We use the configuration icon



to set the data at the **application parameters**
& run



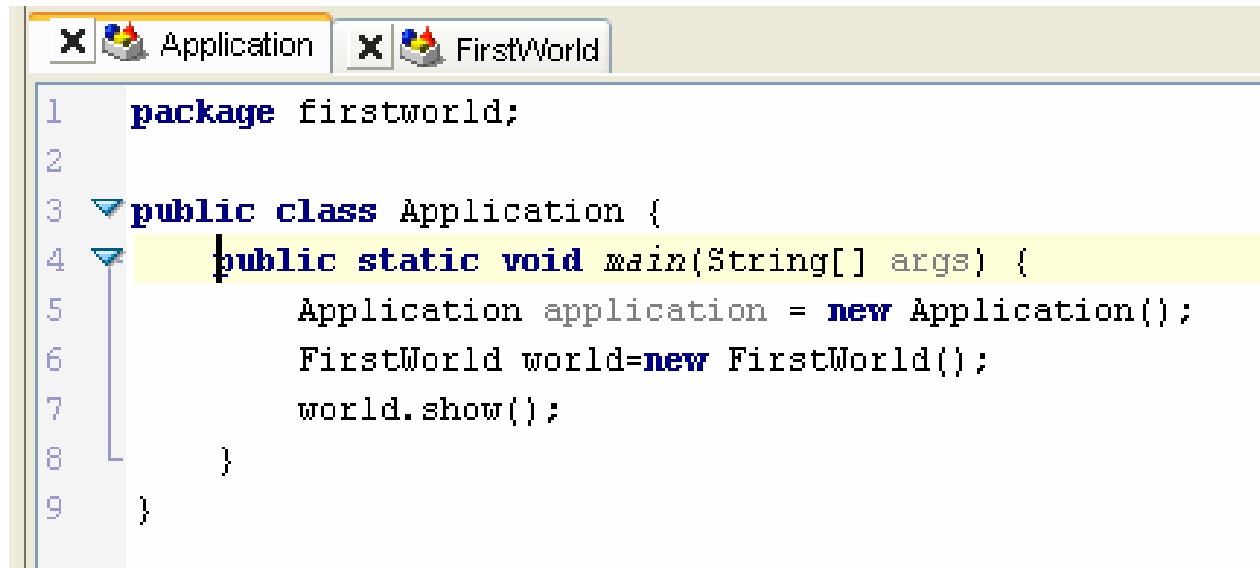
```
BankAccount
1 package omarproject;
2 class BankAccount {
3     // main method
4     public static void main(String[] args) {
5         BankAccount omarAccount = new BankAccount(2000);
6         BankAccount bassamAccount = new BankAccount(1000);
7         omarAccount.deposit(500);
8         System.out.println(" Omar Hamed ");
9         System.out.println(omarAccount.balance);
10        omarAccount.transfer( 345,bassamAccount);
11        System.out.println(" Omar Hamed New Balance ");
12        System.out.println(omarAccount.balance);
13        System.out.println(" Bassam Hamed New Balance");
14        System.out.println(bassamAccount.balance);
15        bassamAccount.setBalance(567);
16        System.out.println(bassamAccount.balance);
17        omarAccount.withdrawal(500);
18        System.out.println(" Omar Hamed New Balance ");
19        System.out.println(omarAccount.balance);
20    }
21    protected int balance;
22    // constructor method
23    public BankAccount (int anAmount) {
24        balance=anAmount;
25    }
```

```

25     }
26     // Methods
27     public int getBalance() {
28         return balance;
29     }
30     public void setBalance (int anAmount) {
31         balance=anAmount;
32     }
33     public void deposit (int anAmount) {
34         balance=balance + anAmount;
35     }
36     public void withdrawal (int anAmount) {
37         if (anAmount <= balance)
38             balance=balance-anAmount;
39         else
40             balance=0;
41     }
42     public void transfer (int anAmount, BankAccount anAccount) {
43         if ( balance >= anAmount){
44             anAccount.deposit(anAmount);
45             balance=balance-anAmount;
46         }
47     }
48 }
49

```

Explore The Following FirstWorld Project



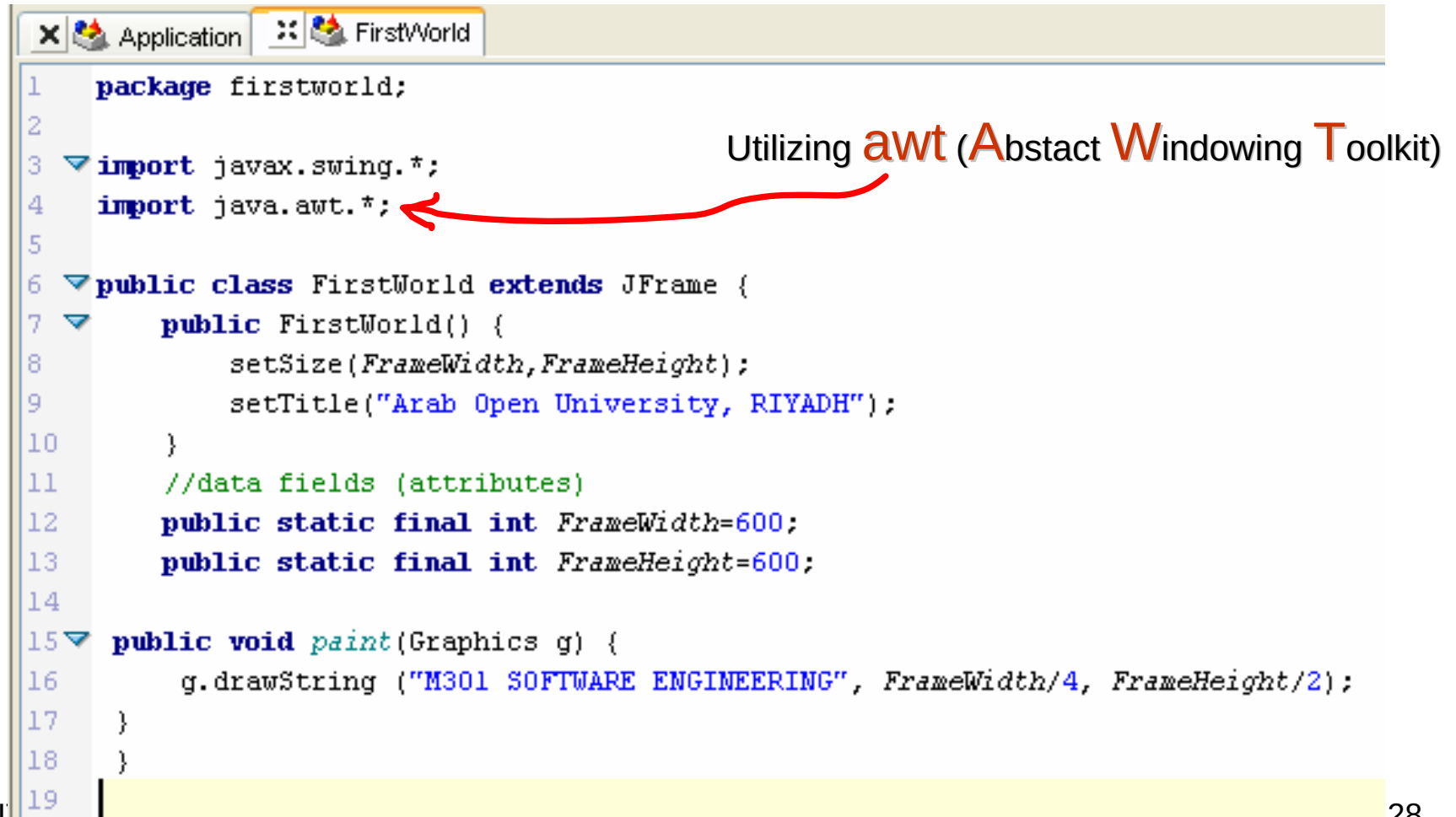
The screenshot shows an IDE window with two tabs: 'Application' and 'FirstWorld'. The 'FirstWorld' tab is active, displaying the following Java code:

```
1  package firstworld;
2
3  public class Application {
4      public static void main(String[] args) {
5          Application application = new Application();
6          FirstWorld world=new FirstWorld();
7          world.show();
8      }
9  }
```

The main method

Explore The Following FirstWorld Project

The FirstWorld class



```
1 package firstworld;
2
3 import javax.swing.*;
4 import java.awt.*;
5
6 public class FirstWorld extends JFrame {
7     public FirstWorld() {
8         setSize(FrameWidth, FrameHeight);
9         setTitle("Arab Open University, RIYADH");
10    }
11    //data fields (attributes)
12    public static final int FrameWidth=600;
13    public static final int FrameHeight=600;
14
15    public void paint(Graphics g) {
16        g.drawString ("M301 SOFTWARE ENGINEERING", FrameWidth/4, FrameHeight/2);
17    }
18 }
19
```

Utilizing **awt** (Abstract Windowing Toolkit)

Exploring FirstWorld Project

The data field `FrameWidth` is declared as

```
Public static final int FrameWidth=600;
```

Public means that the data field can be accessed not only within its class but within other classes too. The **Private** data field can be accessed only within its class.

A **static** data field is a class data field (class variable). There is only one instance of the data field, which is shared by all the objects of the class.

A data field that is declared **final** is a constant and can not be changed.

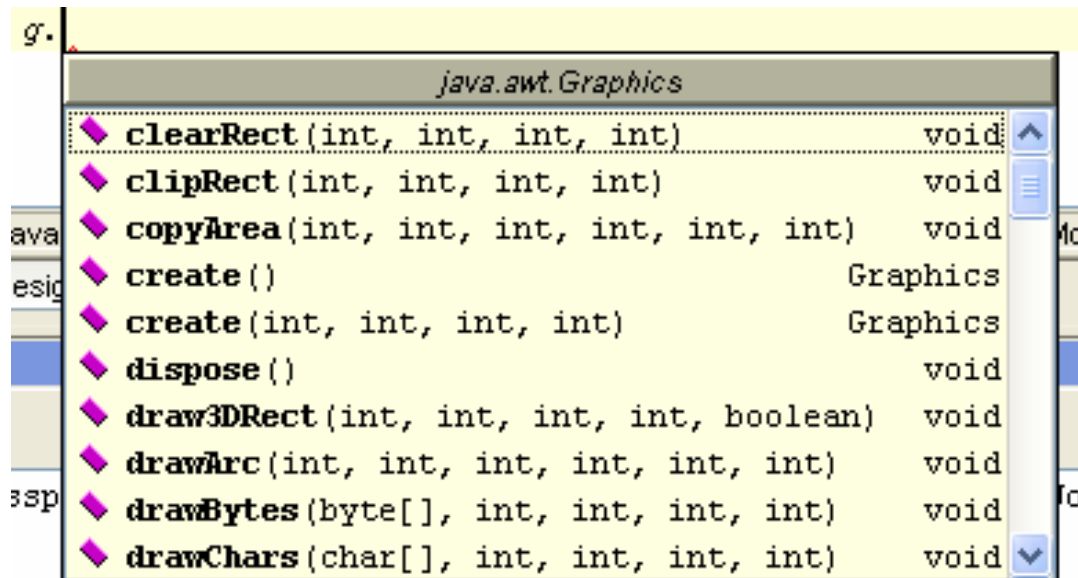
Changing the method **paint** as shown below

What would be the output?

```
15 public void paint(Graphics g) {  
16     int i=0;  
17     g.setColor(Color.RED);  
18     for (i=0; i<10; i++) {  
19         g.drawString("Arab Open University M301",20*i,30*i);  
20     }  
21 }  
22 }
```

The paint method specifies which graphics objects are to be painted on the window frame.

Explore the **paint** method



We can draw ovals, rectangles, polygons, lines, etc.

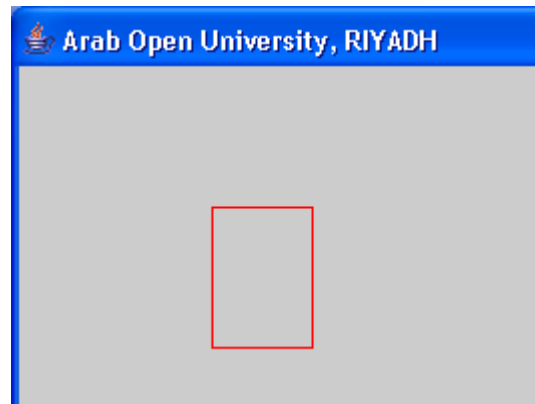
explore !!!!

For example the command

```
g.setColor(java.awt.Color)  
g.drawRect(int x, int y, int width, int height)  
g.drawRect();
```

```
18 g.drawRect(100,100,50,70);  
19
```

**Produced the
rectangle as shown**



Draw other shapes. Be adventurous & artistic.

Explore the modification on our project.

- Explain the work of the **run** method

- Notice the **try** structure

- Notice the **sleep** method

- Notice the data field **xCoord** & **yCoord**

Explore

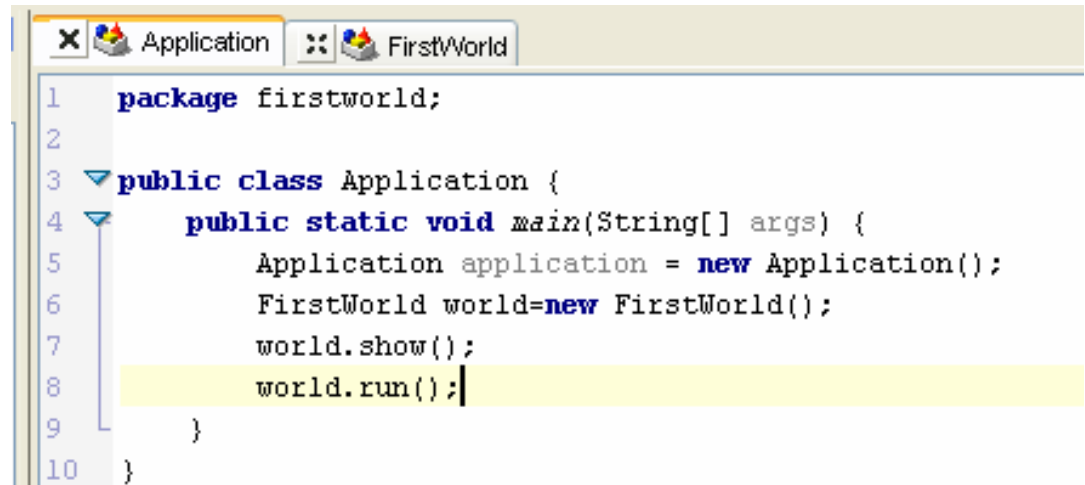
See Block I p. 11 & 12

```
4  import java.awt.*;
5
6  public class FirstWorld extends JFrame {
7      public FirstWorld() {
8          setSize(FrameWidth, FrameHeight);
9          setTitle("Arab Open University, RIYADH");
10     }
11
12     //data fields (attributes)
13     public static final int FrameWidth = 600;
14     public static final int FrameHeight = 600;
15     private int xCoord = 60;
16     private int yCoord = 60;
17
18     public void paint(Graphics g) {
19         super.paint(g); g.setColor(Color.BLUE);
20         g.drawString("Omar Hamed", xCoord, yCoord);
21     }
22
23     public void run() {
24         for (int i = 0; i < 10; i++) {
25             try {
26                 Thread.sleep(2000);
27             }
28
29             catch (Exception e) {
30                 System.exit(0);
31             }
32             xCoord = xCoord + 10;
33             yCoord = yCoord + 10;
34             repaint();
35         }
36     }
```

Explore the modification on our **main** method.

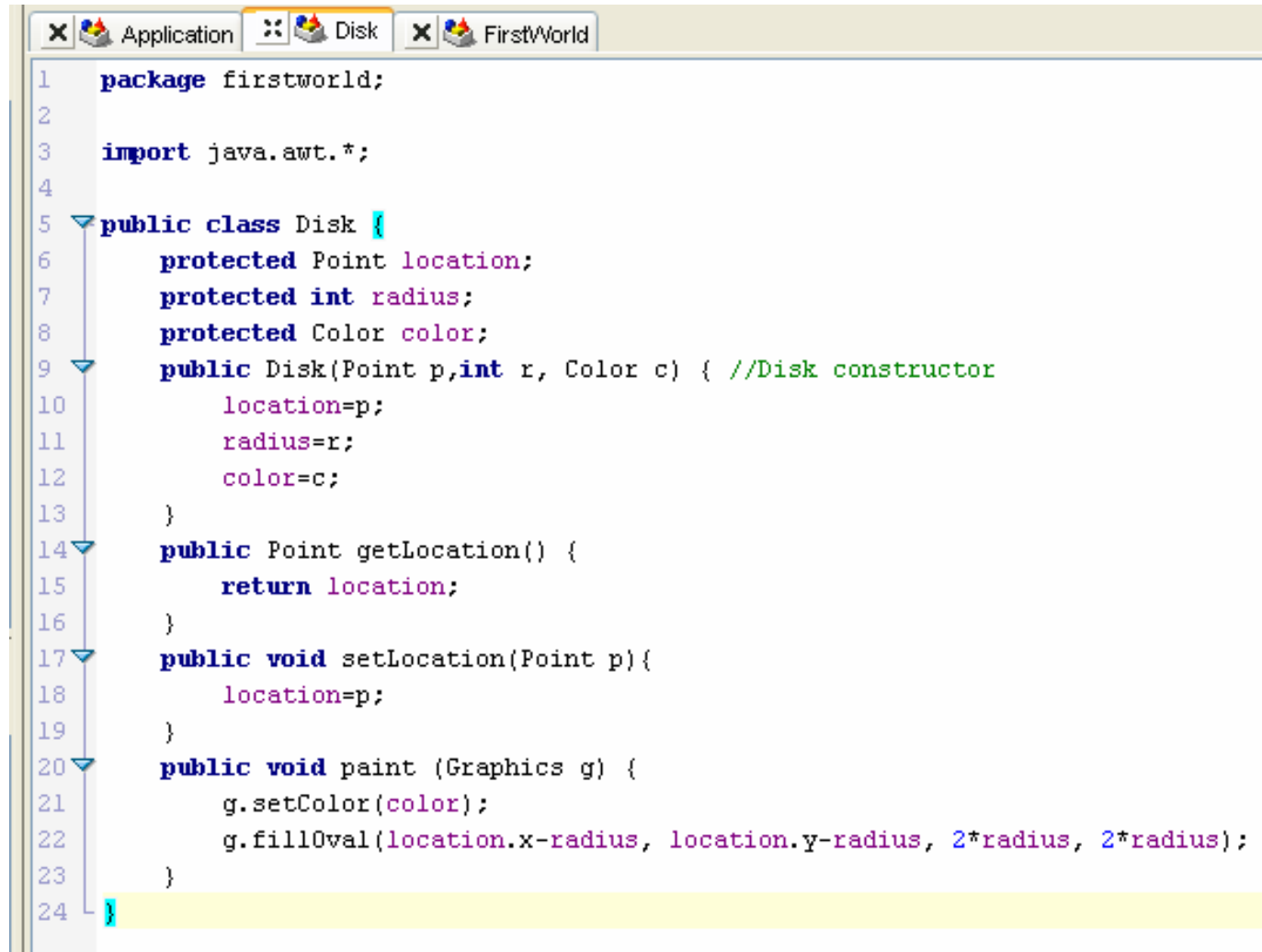
- Explain the work of the **run** method

- What would be the results of running our application? Explore.



```
1 package firstworld;
2
3 public class Application {
4     public static void main(String[] args) {
5         Application application = new Application();
6         FirstWorld world=new FirstWorld();
7         world.show();
8         world.run();
9     }
10 }
```

Explore the Disk class. See pages 14, 15 & 16 of Block I

A screenshot of a Java IDE window with three tabs: 'Application', 'Disk', and 'FirstWorld'. The 'Disk' tab is active, showing the source code for the 'Disk' class. The code is as follows:

```
1 package firstworld;
2
3 import java.awt.*;
4
5 public class Disk {
6     protected Point location;
7     protected int radius;
8     protected Color color;
9     public Disk(Point p,int r, Color c) { //Disk constructor
10         location=p;
11         radius=r;
12         color=c;
13     }
14     public Point getLocation() {
15         return location;
16     }
17     public void setLocation(Point p){
18         location=p;
19     }
20     public void paint (Graphics g) {
21         g.setColor(color);
22         g.fillOval(location.x-radius, location.y-radius, 2*radius, 2*radius);
23     }
24 }
```

Explore the modification. See pages 14, 15 & 16 of Block I

```
1
2 //data fields (attributes)
3 public static final int FrameWidth = 600;
4 public static final int FrameHeight = 600;
5 private int xCoord = 25;
6 private int yCoord = 100;
7 private Point p = new Point(xCoord, yCoord);
8 private Disk d = new Disk(p, 15, Color.BLUE);
9
10 public void paint(Graphics g) {
11     super.paint(g);
12     d.paint(g);
13 }
14
15 public void run() {
16     for (int i = 0; i < 10; i++) {
17         try {
18             Thread.sleep(2000);
19         }
20
21         catch (Exception e) {
22             System.exit(0);
23         }
24         xCoord = xCoord + 25;
25         yCoord = yCoord + 25;
26         p.setLocation(xCoord, yCoord);
27         d.setLocation(p);
28         repaint();
29     }
30 }
31 }
```