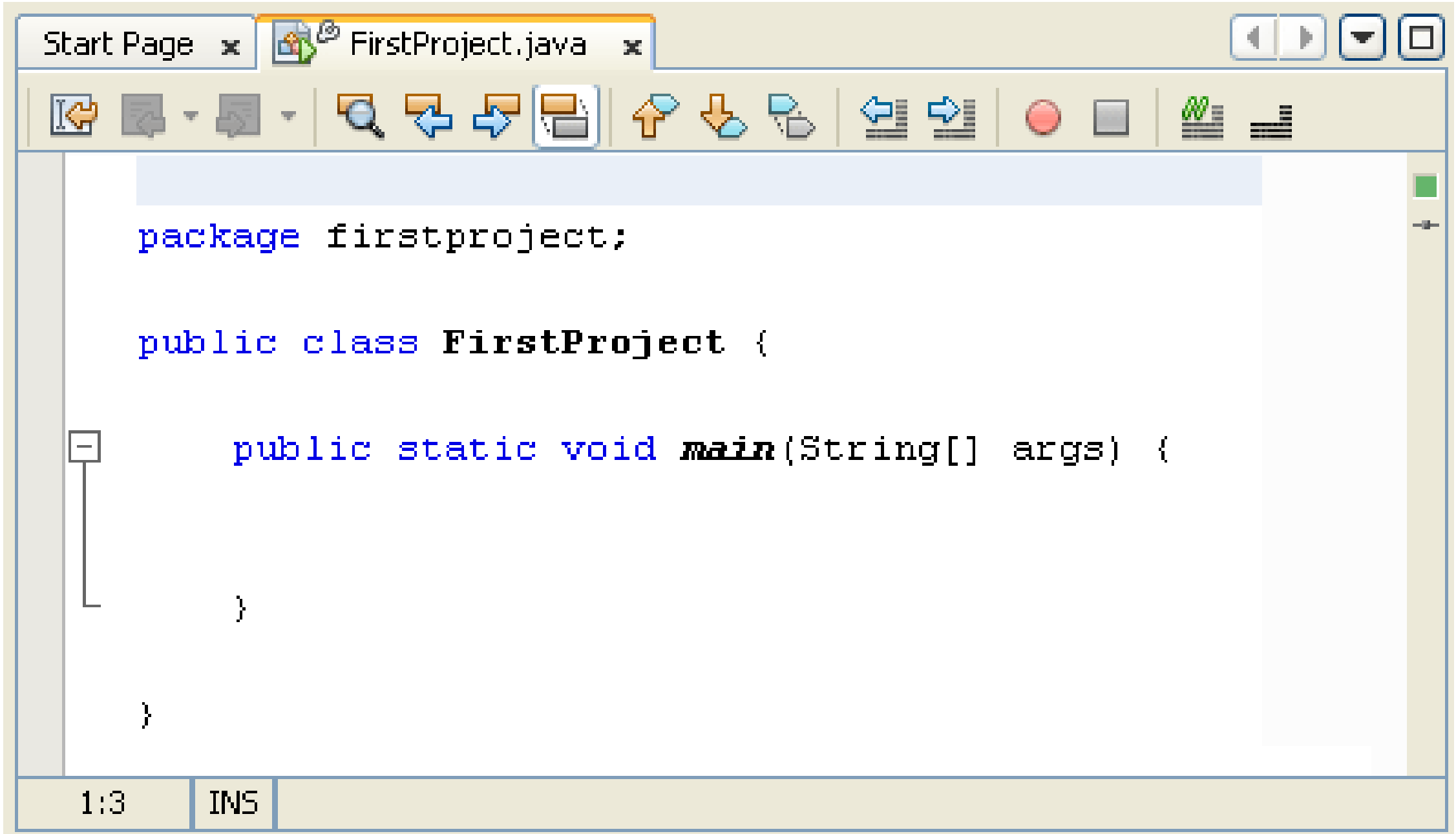
The Structure of Java Code

Java Identifiers

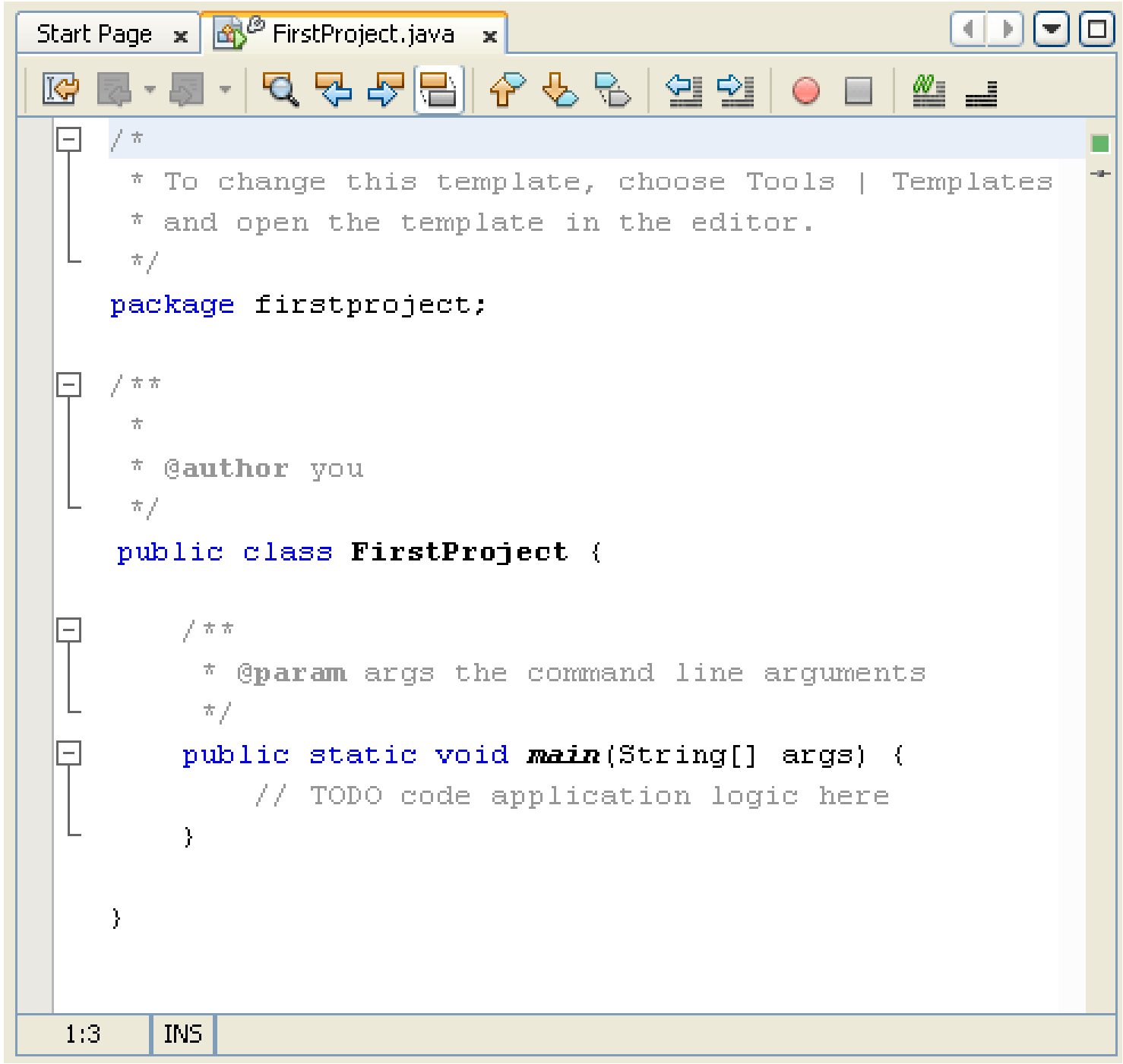
- All Java components require names. Names used for classes, variables and methods are called identifiers. In Java, there are several points to remember about identifiers. They are as follows:
- All identifiers should begin with a letter (A to Z or a to z), currency character ($) or an underscore (_).
- After the first character, identifiers can have any combination of characters.
- A keyword cannot be used as an identifier.
- Most importantly identifiers are case sensitive.
- Examples of legal identifiers: age, $salary, _value, __1_value
- Examples of illegal identifiers: 123abc, -salary


Start Page
FirstProject.java
package firstproject;
public class FirstProject {
public static void main(String[] args) {
}
}
1:3
INS

- You can see we have the package name first. Notice how the line ends with a semicolon. If you miss the semicolon out, the program won't compile:
  - package **firstproject;** 
- The class name comes next:
  - public class **FirstProject {**
  - **}**
- You have to tell Java where code segments start and end.
- You do this with curly brackets. { }
- The start of a code segment is done with a left curly bracket { and is ended with a right curly bracket }.
- Anything inside of the left and right curly brackets belong to that code segment.
- What's inside of the left and right curly brackets for the class is another code segment. This one:
  - public static void ***main(*** String[ ] args **)** {
  - };

Java Comments

- Java comments allow you to write notes to yourself and others working on the same project.
- You can type whatever you want inside of your comments.
- When the program runs, comments are ignored.
- However, it's usual to have comments that explain what is you're trying to do.


Start Page
FirstProject.java
/*
* To change this template, choose Tools | Templates
* and open the template in the editor.
*/
package firstproject;
/**
*
* @author you
*/
public class FirstProject {
/**
* @param args the command line arguments
*/
public static void main(String[] args) {
// TODO code application logic here
}
}
1:3
INS

- You can have a single line comment by typing two slashes, followed by your comment:
  - **//This is a single line comment**
- If you want to have more than one line, you can either do the following:
  - **//This is a comment spreading**
    //over two lines or more
- Or you can do this:
  - **/***
    This is a comment spreading
    over two lines or more
    because it's long
    ***/**
- In the comment above, we starts with **/***. To end the comment, we have ***/** instead.

The Output Window

- To output text to a console window. We Add the following line to our main method:

```
        System.out.println("");
    }
```

- Once you have your double quotes in place, type your text:

```
        System.out.println("My First Project");
    }
```

- public static void **main(** String[ ] args **) {**
- **System.out.println("My First Project");**
- **}**

Java Reserved Keywords

abstract	assert	boolean	break
byte	case	catch	char
class	const	continue	default
do	double	else	enum
extends	final	finally	float
for	goto	if	implements
import	instanceof	int	interface
long	native	new	package
private	protected	public	return
short	static	strictfp	super
switch	synchronized	this	throw
throws	transient	try	volatile
void	while		