Session 3

Working with NetBeans IDE

Objectives:

- Showing how IDE software facilitates the process of programming.
- Introduce the basics of NetBeans IDE.
- Introducing debugging and its importance.
- Introducing error types in Java and understanding error messages
- Discriminating between run-time (semantic) errors and compile-time (syntax) errors.

3.1 IDE Software

IDE stands for Integrated Development Environment, IDE is a software that helps us to write, compile, run and debug programs. IDE programs vary and each one of them usually targeted to certain programming language, the following table shows well-known IDE programs:

Programming Languages	IDE
Java	• NetBeans • Forte for Java • Borland JBuilder
C/C++	• Microsoft Visual Studio • Borland C++
PHP/JSP/ASP	• Macromedia Dreamweaver
ASP.NET	• Microsoft Visual Studio .NET • Sharp Develop

We can notice from table above that we have various brands of IDEs, since we are interested in Java programming language in this course, we will focus on its IDEs. NetBeans is the best offered IDE for us as Java learners, because it is free compared to JBuilder, more advanced and stable compared to Forte. So lets start having a look at NetBeans.

3.2 NetBeans IDE

Figure 3.1 shows the main screen of NetBeans IDE (version 5.0). You can download NetBeans from http://java.sun.com, Once you have JDK 1.5 installed, NetBeans installation is a straight-forward process.

To use NetBeans for writing and running your Java programs, Follow these steps:

1. Create a new project by going to **File > New Project**, or by clicking on **New Project button** on the toolbar; all programs written using NetBeans must be included in projects to be able to be compiled and run. Project creation is a simple process consists of two steps. First step is defining **project category** and **type**. In our case, we leave the default settings, that is, **General** category and **Java Application** type. Second step is specifying **project name** and **main class**.

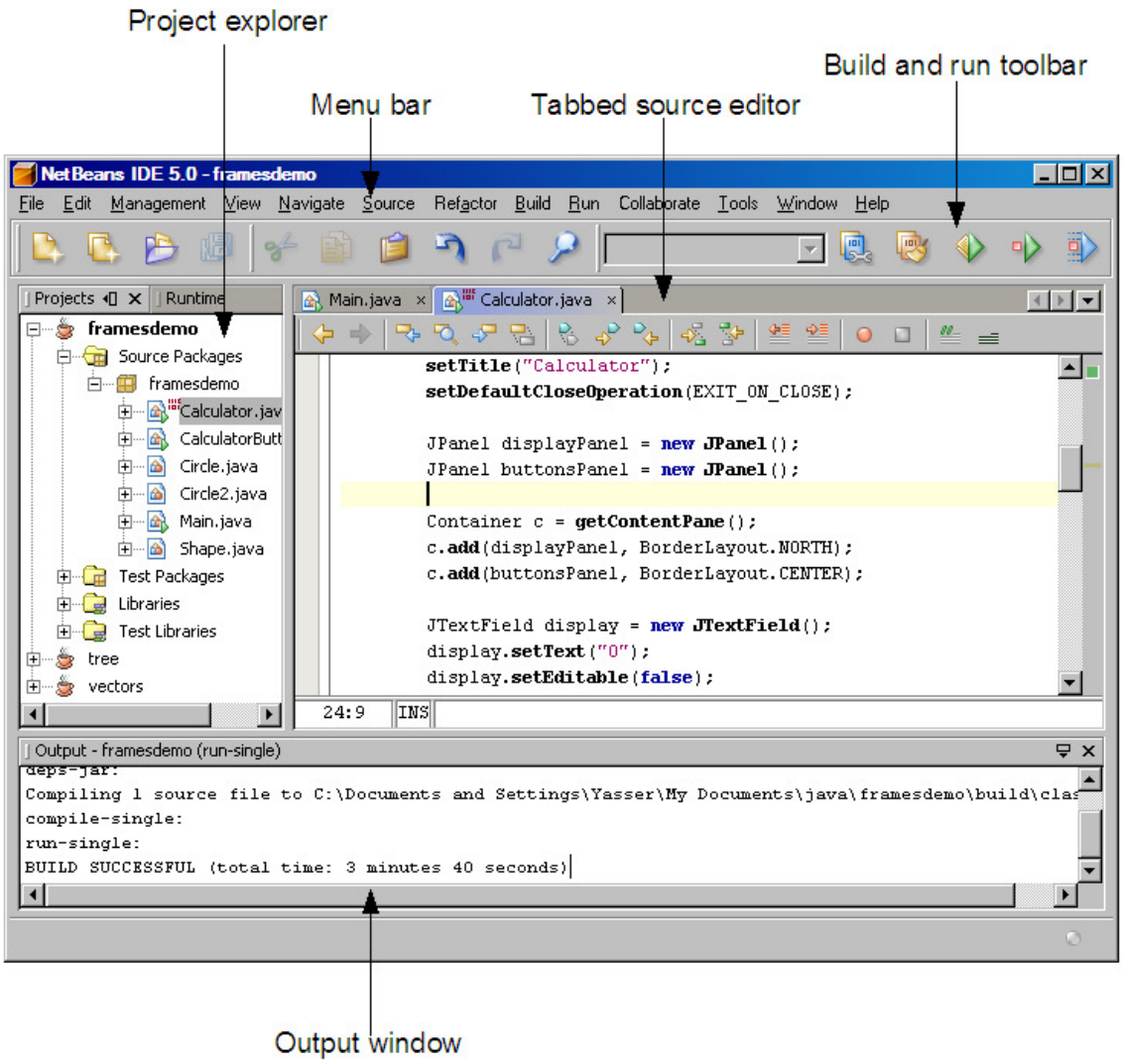

Figure 3.1: NetBeans IDE main screen

2. After creating a project, you are ready to start coding, on the **project explorer** you will see a tree that has the project name you specified as root. Project is split into four folders: **Source packages**, **Test packages**, **Libraries** and **Test libraries**. **Source packages** is the folder containing the source code files (.java files) and hence it is the folder we are interested in. Source code is usually split into packages, so we can group classes of related functionality together. Once you create a project, a default package that holds project name is created.
3. In the **code editor**, you will see the created `Main` class, with its `main` method already coded, so you need just to start programming.
4. In the `main` method, try to write some statements you've learned from session 1, after you are done, you may compile and run the program by selecting **Run > Run main project**, pressing F6 key on the keyboard, or clicking **Run button** on the toolbar 
5. The **output window** appears at the bottom of the IDE and shows the outputs of compilation and running processes.

3.3 Debugging Your Program

The importance of debugging arises from the ability to trace the code statement by statement and monitor values of variables at every moment. NetBeans allows you to debug your program and trace any run time errors that may exist. Debugging in NetBeans IDE means that you define one or more *break points* in your code, break point is a sign you may put on any line of your program to have the debugger stop at it, when the execution of the program reaches a break point, it hangs allowing you to review the values of variables at that moment. To add a break point, click on the left of the line you want to insert the break point in. Figure 3.2 shows a break point added in some lines.

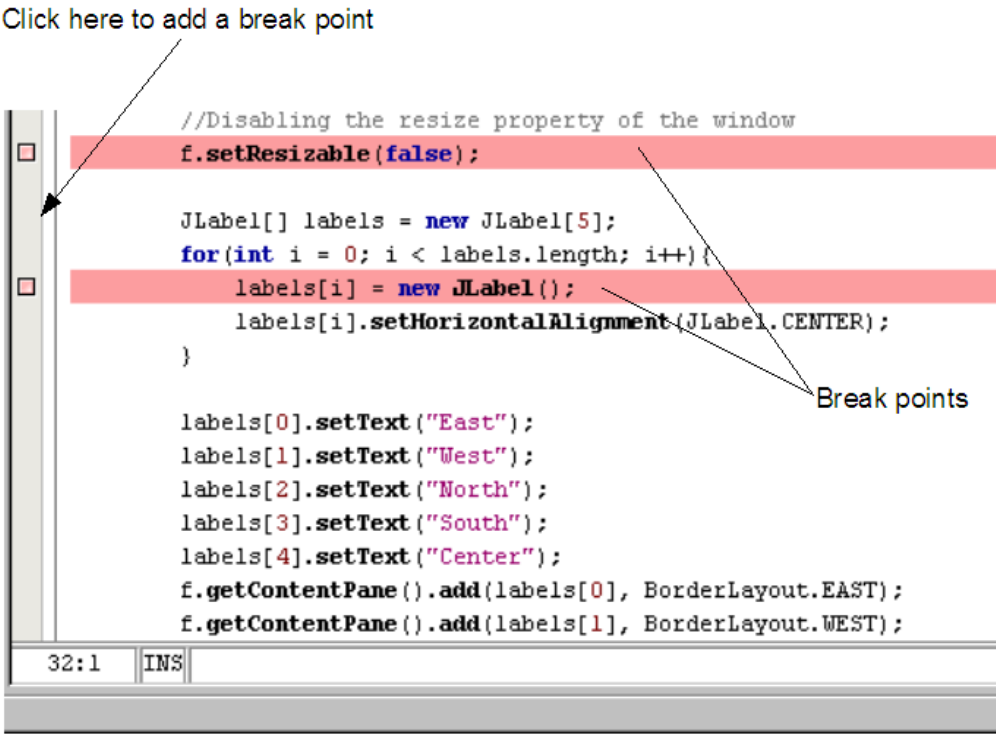

Figure 3.2: Break points

To show how debugging works, make a small test by following the procedure below:

1. Write the `HelloWorld` program from session 1.
2. Add a break point in the second line (`System.out.println("How r u 2day?");`).
3. Debug the program by pressing F5 key on the keyboard or by clicking on **Debug button** on the toolbar.
4. Observe the output of the program, what do you notice?
5. Notice that when debugger reaches a break point, it turns to green, this means that this is the next statement will be executed, as we put a break point on the second line, the statement is not executed until we select to *continue*, this can be done by pressing Ctrl +F5 or by clicking on C**ontinue button** on the toolbar.
6. To stop debugging, press Sift+F5 on the keyboard, or click on **Stop button** on the toolbar.

3.4 Error Types in Java

In Java (and other programming languages) there are two major types of errors:

- Compile-Time errors (Syntax errors).

- Run-Time errors, these errors fall into two categories: exceptions and semantic errors.

Compile time errors are those the compiler detects while translating your program from Java language to machine language, the most common reason for these errors is there is something in your code does not follow Java syntax (that's why we call them syntax errors). The following examples show some of these errors:

```
1. System.out.println("Hellow")
2. system.out.println("Hello");
3. System.out.println("Hello);
4. class Hello World{
5. Class HelloWorld{
6. public Static void main(string[] args){
```

OK, let's start with statement 1, what is wrong with it? Clearly Hello don't have "w" at the end of it! However, this is not the error we are looking for, we are looking for violation of Java language rules not English language rules. As we know, each statement in Java should be terminated with semicolon "`;`" which is absent in our case. The message the compiler will generate in this case is "`';' expected`", error message also includes the line in which error exists as shown in figure 3.3.

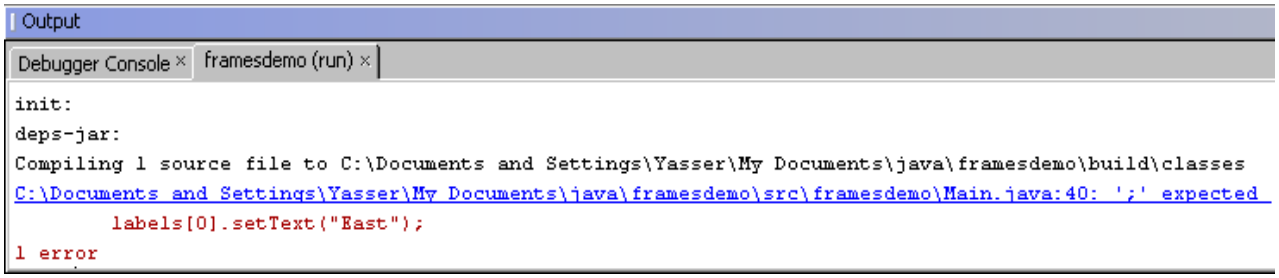

Figure 3.3: Compile-Time error message.

In statement 2, you can notice that `system` is written in lower case, as we've learned before, Java is case sensitive, and as `System` is a class in Java, it starts with upper case letter. The message you will see in this case is "`Cannot find symbol`". This message means the keyword or member name you are calling is not recognized as valid one. The most common reason for this error is a spelling mistake.

In statement 3 , the error message is "`unclosed string literal`", because the opening double quotation (") does not have a matching closing one.

In statement 4, the error is that the given class name is not valid, as class name cannot have spaces according to naming conventions in Java (as we shall see in next session). The error message in this case is "`'{' expected`", the compiler assumes that class name finishes when it finds space, after the space it is expecting the opening bracket "`{`".

In statement 5, the message generated is "`class or interface expected`", it is clear that `class` keyword must start with lower case letter as all keywords in Java do so.

In statement 6, the message generated is "`<identifier> expected`", because `Static` is not a valid keyword. Notice that in this statements we have another error which is "`cannot find symbol`" because `string` is written mistakenly in lower case, but it will not be detected until first error is fixed because it appears first.

Run-Time errors appear when program run, the compiler cannot detect them because there are no errors or rules violations. First category of run-time errors is *exceptions*, exception is a non expected error happens while executing a code, an example of exceptions is

`ArithmeticException` which occurs when you try to divide by zero. The second category are *semantic* errors, those are not actual errors detected by compiler, nor exceptions occur at runtime, but they are unexpected results or outputs of your program, for example, if your program is supposed to calculate the sum of three numbers, and we give it 1, 2 and 3 as inputs, its output should be 6, otherwise we have a semantic error.
NetBeans can detect syntax errors while you type your program, so you can find them before compiling you program. The statement containing an error is underlined by red line, and you can see the error message by positioning the mouse pointer over the line as shown in figure 3.4. We will have a closer look at run-time errors and how to deal with them in the coming sessions.

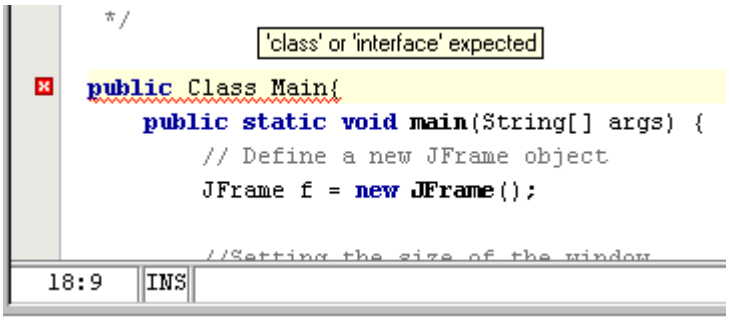

Figure 3.4: Error detection in NetBeans

Exercises

1. Create a project using NetBeans called MyFirstProject, and type a program to print your name and student ID each in separate line using only one `println` statement.
2. What is the error message expected from each of the following statements? (Also show if there is more than one error, and mention which one is detected first)

```
1. System.out.println(Hello World!);
2. System.out.pint("How are you?");
3. public static void main(String[] args);{
4. class My Class{
5. public static void main(String() args){
6. System.out.Print ln("Hi man!");
7. System.out.print('What's up');
8. System.out.println("I wonder where is the error?\m");
```

Lab Extras

You can add a new class to your default package and make it the default class to be run instead of `Main` class. To do this, right-click package in the **project explorer**, open **Add** menu and select **Java Class** from the list. Enter new class name and click **OK** to see your newly added class. Note that the Main class is still open and you can navigate between two class using tabs above **source editor**.
To make your new class the default one (that is, it will run when you run the project), right-click the project icon (Java cup) in the **project explorer** and select **Properties** from the menu. This will open up **Project Properties** dialog. From the tree on the left, select **Run** category, and type your class name in place of Main in **Main Class** text field. Be ware not to delete package name mistakenly.
Now type a main method in your new class and put some code in it, run the project and you will see the result. You may also run a non-default class by right-clicking it in the project explorer and selecting run or using Shift + F6 keys while opening its code in the editor.