# Introduction to Java

Sayed Ahmad Sahim

Benawa University

September 14, 2017

# Road Map

1. Introduction to Java
   - Comments
   - The main() Method
2. Integrated Development Environment IDE
3. Data types and variable
4. END

# What is Java?

# Introduction to Java

Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible.(Gosling, James)

# Java Editions

- Java Platform, **Standard Edition** or Java SE is a widely used computing platform for development and deployment of portable code for desktop and server environments.
- Java Platform, **Enterprise Edition** (Java EE), formerly known as Java 2 Platform, Enterprise Edition (J2EE), is a computing platform for development and deployment of enterprise software (network and web services).
- Java Platform, **Micro Edition** or Java ME is a computing platform for development and deployment of portable code for embedded and mobile devices (micro-controllers, sensors, gateways, mobile phones, personal digital assistants, TV set-top boxes, printers).

# How to get a working environment for Java?

# JVM

- JVM is an abstract computing machine, having an instruction set that uses memory. Virtual machines are often used to implement a programming language.
- JVM is the cornerstone of the Java programming language. It is responsible for Java's cross-platform portability and the small size of its compiled code.

- The Java Development Kit (JDK) is a software development environment used for developing Java applications and applets.
- It includes:
  - the Java Runtime Environment (JRE)
  - an interpreter/loader (java)
  - a compiler (javac)
  - an archiver (jar)
  - a documentation generator (javadoc)
  - other tools needed in Java development

# Syntax of a Language

The set of rules you must follow when writing program code are referred to as syntax. in the first java hello world program we can see the following.

# Syntax of Java

Code you saw in first java hello world program included the following parts

- Comments start with $/* \ */$ statements inside these signs are ignored by compiler
- The next line of the code was *public class HelloWorld* You must define a class in every Java program
- Class is a group of functions and characteristics in Java code that can be reused.

## Continue

- Methods are groups of statements that are executed by your computer when instructed to do so.
- The next line of code starts the main() method definition *public static void main(String args[])*
- The next line of the code instructs computer to print Hello, world on the screen *System.out.println("Hello, world!");*
- *semicolon ;* are used to end statements like we use full stop in our languages to end a statement.
- The last two lines of code are the closing braces for the main() method and the HelloWorld class definition.

# Comments

Adding comments to your code is not required, although it is definitely a good practice. Comments help you or anyone reading your code understand what your program does. Comments are not executed by the compiler.

We have two types of comment in java.

- Single Line starts with // until the end of the line.
- Multi Line comment starts /* */

# The main() Method

When running a Java application, the main() method is the first thing the interpreter looks to. It acts as a starting point for your program and continues to drive it until it completes.
Every Java application requires a main() method or it will not run *public static void main(string args[])*.

- public makes this method accessible from other classes
- static ensures that there is only one reference to this method used by every instance of this program (class)
- main is the name of the method
- in front of the main method inside parenthesis are the arguments which the method accepts.
- String args[] Specifically, it is an array (list) of command-line arguments

# Integrated Development Environment IDE

An integrated development environment (IDE) is a programming environment that has been packaged as an application program, typically consisting of a code editor, a compiler, a debugger, and a graphical user interface (GUI) builder.

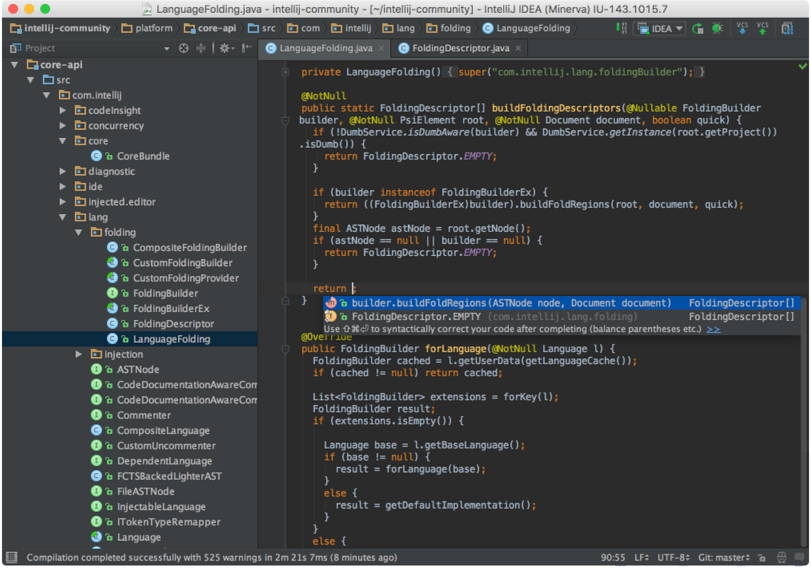
# Question

Why should we use IDE?

## Benefits of using IDE

- Compiler included with IDE
- Text editor is also Included with IDE
- Error checking
- Code navigation
- Code completion
- Code generation
- Code coloring
- Debugging

# Intellij IDEA



```java
private LanguageFolding() { super("com.intellij.lang.foldingBuilder"); }

@NotNull
public static FoldingDescriptor[] buildFoldingDescriptors(@Nullable FoldingBuilder
builder, @NotNull PsiElement root, @NotNull Document document, boolean quick) {
  if (!DumbService.isDumbAware(builder) && DumbService.getInstance(root.getProject())
.isDumb()) {
    return FoldingDescriptor.EMPTY;
  }

  if (builder instanceof FoldingBuilderEx) {
    return ((FoldingBuilderEx)builder).buildFoldRegions(root, document, quick);
  }
  final ASTNode astNode = root.getNode();
  if (astNode == null || builder == null) {
    return FoldingDescriptor.EMPTY;
  }

  return ;
}
```

```
builder.buildFoldRegions(ASTNode node, Document document)    FoldingDescriptor[]
FoldingDescriptor.EMPTY (com.intellij.lang.folding)          FoldingDescriptor[]
```

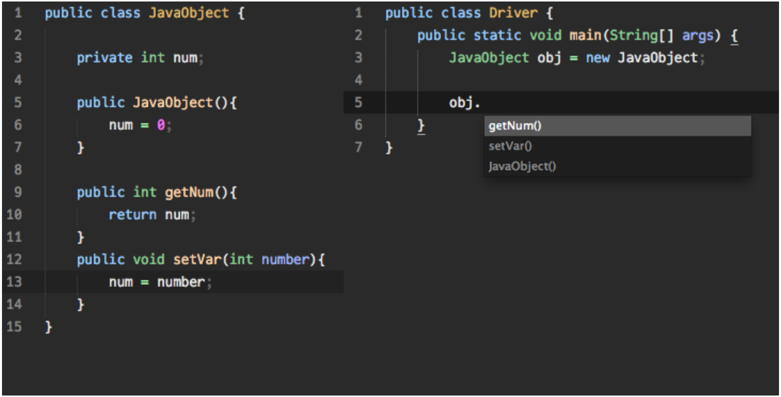
Use ⇧⌘⏎ to syntactically correct your code after completing (balance parentheses etc.) >>

```java
@Override
public FoldingBuilder forLanguage(@NotNull Language l) {
  FoldingBuilder cached = l.getUserData(getLanguageCache());
  if (cached != null) return cached;

  List<FoldingBuilder> extensions = forKey(l);
  FoldingBuilder result;
  if (extensions.isEmpty()) {

    Language base = l.getBaseLanguage();
    if (base != null) {
      result = forLanguage(base);
    }
    else {
      result = getDefaultImplementation();
    }
  }
  else {
```

Compilation completed successfully with 525 warnings in 2m 21s 7ms (8 minutes ago)

90:55 LF÷ UTF-8÷ Git: master÷

# Sublime Text

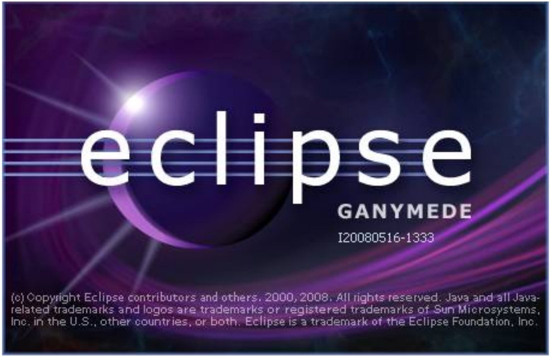


```
public class JavaObject {

    private int num;

    public JavaObject(){
        num = 0;
    }

    public int getNum(){
        return num;
    }
    public void setVar(int number){
        num = number;
    }
}
```

```
public class Driver {
    public static void main(String[] args) {
        JavaObject obj = new JavaObject;

        obj.
    }
}
```

# Eclipse

Eclipse is a Java-based open source platform that allows a software developer to create a customized development environment (IDE) from plug-in components built by Eclipse members. Eclipse is managed and directed by the Eclipse.org Consortium.



(c) Copyright Eclipse contributors and others. 2000, 2008. All rights reserved. Java and all Java-related trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S., other countries, or both. Eclipse is a trademark of the Eclipse Foundation, Inc.

# NetBeans

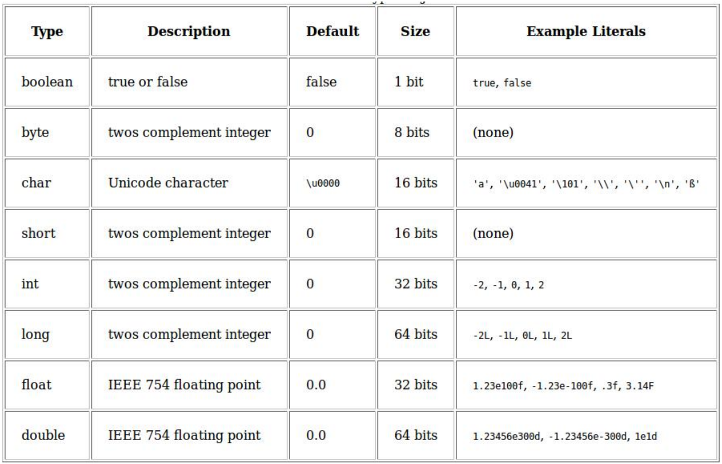
NetBeans is an open-source integrated development environment (IDE) for developing with Java, PHP, C++, and other programming languages

- Open source Free IDE

# Data types

- A data type is a set of values and a set of operations defined on them.
- For example, we are familiar with numbers and with operations defined on them such as addition and multiplication.
- There are two different types of variable:
  - Primitive/Builtin Data types
  - Referenced or User defined data types

# Primitive Data type



| Type | Description | Default | Size | Example Literals |
|---|---|---|---|---|
| boolean | true or false | false | 1 bit | true, false |
| byte | twos complement integer | 0 | 8 bits | (none) |
| char | Unicode character | \u0000 | 16 bits | 'a', '\u0041', '\101', '\\', '\'', '\n', 'ß' |
| short | twos complement integer | 0 | 16 bits | (none) |
| int | twos complement integer | 0 | 32 bits | -2, -1, 0, 1, 2 |
| long | twos complement integer | 0 | 64 bits | -2L, -1L, 0L, 1L, 2L |
| float | IEEE 754 floating point | 0.0 | 32 bits | 1.23e100f, -1.23e-100f, .3f, 3.14F |
| double | IEEE 754 floating point | 0.0 | 64 bits | 1.23456e300d, -1.23456e-300d, 1e1d |

## Java Tokens

- A token is the smallest element in a program that is meaningful to the compiler. These tokens define the structure of the language.
- The Java token set can be divided into five categories:
  - Identifiers
  - Keywords
  - Literals
  - Operators
  - Separators.

# Variables

- Instance Variables (Non-Static Fields)
- Class Variables (Static Fields)
- Local Variables. A method stores its temporary state in
- local variables.
- Parameters. They are the variables that are passed to the methods of a class.

# Variable Declaration

- Identifiers are the names of variables.
- They must be composed of only letters, numbers, the underscore, and the dollar sign ($). They cannot contain white spaces.
- Identifiers may only begin with a letter, the underscore, or the dollar sign. A variable cannot begin with a number.
- All variable names are case sensitive.
- **Syntax** *datatype1 variable1, datatype2 variable2, datatypen variablen;*
- **Initialisation** *Variablename* = *value;*

# Arrays

An array is a group of variables that share the same data type, and are referred to by a common name. Arrays of any type can be created and may have one or more dimensions.

# Declaring Array Variables

- Arrays are indexed based data structures
- Size should be defined during declaration
- *Datatype[] nameīnitialization*