Java 11 and Beyond
MAKING ECLIPSE JDT FUTURE–READY

MANOJ PALAT – IBM
@manojnp
Java Releases – Road Traveled++

Version
- 1.0
- 1.1, 1.2, 1.3, 1.4, 1.5, 1.6
- 1.7
- 1.8
- 9
- 10
- 11
- 12
- 13...

Release Date
- 1996
- 2011 (String Switch, t-w-r, dyn-support)
- 2014 (Lambda, Null)
- Sep 2017 (Modules,
- Mar 2018
- Sep 2018
- Mar 2019
- Sep 2019...
Eclipse/JDT

JAVA DEVELOPMENT TOOLS (JDT) UI

JAVA MODEL
- Workspace
- Projects
- CU, Type, Methods
- Lightweight - views

DOM AST
- Compilation Unit
- Fine grained
- Statements, identifiers
- Fully resolved

JAVA SEARCH
- Declarations
- References
- Hierarchy
- Java Constructs

ECLIPSE JAVA COMPILER

Batch Compiler
Agenda

- Past
  - Java 7 & 8 (LE, indy)
  - Java 9 (Modules)
- Present
  - Java 10 (Local Variable Type Inference)
  - Java 11 (var in LE, condy, Nesting)
- Future
  - Java 12 (Raw String, Switch Expressions)
  - And Beyond...
Relevant Historical Features - 1.7, 1.8, 9

- `Java.lang.invoke.MethodHandle, MethodType` (1.7)
- `CONSTANT_MethodHandle_info, CONSTANT_MethodType_info` (1.7)
- `invokedynamic` (1.7)
- `Lambda Expressions` (1.8)
- `_` (Underscore) valid identifier (1.7) warning (1.8) Invalid identifier (9)
- `Modules` (9)
- `t-w-r` with effectively final variables (9)
Current Features - 10, 11

- Local variable type inference (10)
- `var` as lambda parameter (11)
- `Constant_Dynamic`, ldc enablement.
- Nesting
- --enable-preview
Local Variable Type Inference - 10, 11

- Avoid boilerplate code, less ceremony
- Var allowed in
  - variable declaration with initializers,
  - index variable for loop
  - enhanced-for loop
  - lambda expression parameter (11)
- Null init or no init disallowed
- Var not a keyword
- Var is a reserved type
  - `var var = 10;` allowed
Constant_Dynamic (11)

- Reduce the cost and disruption of creating new constants
- Allows flexibility
- Essentially provide a new constant pool with user specs
- Using the existing bootstrap mechanism of LE to this end
- Not utilized by Java as of 11 (initial use expected by 12)
Nest (11)

- JLS allows unrestricted access within a top level type
- private access between nestmates is not permitted by the JVM access rules – compiled into different classes
- Enter bridge method with package private promotion
- Nestmates – Nest Host, Nest Member(s)
Preview Feature

- Language/vm feature
- Fully specified
- Fully implemented
- Not experimental
- Impermanent
- To invoke feedback
- *May* become permanent
- Unavailable by default at compile time and run-time
- **--enable-preview**
Raw String Literals (12) - preview

- span multiple lines of source code and does not interpret escape sequences, such as \n, or Unicode escapes, of the form \uXXXX.
- `Runtime.getRuntime().exec("C:\Program Files\foo" bar`);
- `Runtime.getRuntime().exec(`"C:\Program Files\foo" bar`);`
- `System.out.println("this".matches("\\w\\w\\w\\w`))
- `System.out.println("this".matches(`\\w\\w\\w`)`)

EclipseCon Europe 2018
Raw String Literal Examples (12)

- `""` // a string containing " alone
- ```can`t``` // a string containing 'c', 'a', 'n', '"' and 't'
- `This is a string` // a string containing 16 characters
- `
` // a string containing '\' and 'n'
- `\u2022` // a string containing '\', 'u', '2', '0', '2' and '2'
- `This is a` two-line string` // a single string constant
Switch Expressions (12) - preview

- Issues/Irritants:
  - Default control flow: Fall through
  - Default scoping (Single scope) – one monolithic
  - Finally, a statement

- Extend the switch statement so that it can be used as either a statement or an expression, and that both forms can use either a "traditional" or "simplified" scoping and control flow behavior.
Switch Expressions (325)

```java
switch (day) {
    case MONDAY:
        System.out.println(6);
        break;
    case TUESDAY:
        System.out.println(7);
        break;
    case THURSDAY:
        System.out.println(8);
        break;
    case WEDNESDAY:
        System.out.println(9);
        break;
}
```
int numLetters;
switch (day) {
case MONDAY:
case FRIDAY:
case SUNDAY:
    numLetters = 6;
    break;
case TUESDAY:
    numLetters = 7;
    break;
case THURSDAY:
case SATURDAY:
    numLetters = 8;
    break;
case WEDNESDAY:
    numLetters = 9;
    break;
default:
    numLetters = 100;
}
References

- **Java 12 Page (Official):**
  - [http://openjdk.java.net/projects/jdk/12/](http://openjdk.java.net/projects/jdk/12/)

- **Eclipse JDT Java Plan / Wiki (Mapping, Entry Points and info):**
  - [https://wiki.eclipse.org/JDT_Core/Plan/Java](https://wiki.eclipse.org/JDT_Core/Plan/Java)

- **Top Level Java 12 Support bug Eclipse JDT**
  - [https://bugs.eclipse.org/bugs/show_bug.cgi?id=536055](https://bugs.eclipse.org/bugs/show_bug.cgi?id=536055)

- **BETA_Java_12 branch**
  - **JEP 325**: [http://openjdk.java.net/jeps/325](http://openjdk.java.net/jeps/325)
  - **JEP 326**: [http://openjdk.java.net/jeps/326](http://openjdk.java.net/jeps/326)
A pattern is a combination of a predicate that can be applied to a target, along with a set of binding variables that are extracted from the target if the predicate applies to it.

```java
if (x instanceof Integer) {
    int intValue = ((Integer) obj).intValue();
    // use intValue
}
```

- test (x is an integer)
- Conversion (cast)
- Destructuring (extract)

```java
if (x matches Integer i) {
    // can use i here
}
```

- Switch Expressions
- Dynamic Constant
Thank You

@manojnp

https://in.linkedin.com/in/manojnp

EclipseCon Europe 2018