Java features you may have missed

Diamond operator

Instead of the standard:

```
ArrayList<String> al = new ArrayList<String>();
```

You can drop the second qualifier:

```
ArrayList<String> al = new ArrayList<>();
```

Underscores in Large Numeric Values

Instead of the less readable numeric value like:

```
long largeNumber =  1231321326;
```

You can have a value separated by underscores:

```
long largeNumber =  1_231_321_326;
```

Inline multiple catch expressions

Instead of:

```
try
{
    callBusinessMethod();

} catch (NumberFormatException ex)
{
    handleError(ex);

} catch (IOException ex)
{
    handleError(ex);
}
```

You can have a simplified form, like:

```
try
{
    callBusinessMethod();

} catch (NumberFormatException | IOException ex)
{
    handleError(ex);

}
```

Autocloseable interface

Object extending this interface will not need to be explicitly close. This is increasingly important with for database operations from java and includes : *java.sql.CallableStatement, Connection, PreparedStatement, Statement, ResultSet.*

```
Statement stmt = null;
try {
  stmt = con.createStatement();
} catch (Exception ignore) {
} finally {
  if (stmt != null) stmt.close()
}
```

It can be simplified:

```
try (Statement stmt = con.createStatement()) {
} catch (Exception ignore) {
}
```

Effectively final

Running the code below in Java 7, without the final qualifier will throw an error: "*Cannot refer to the non-final local variable message defined in an enclosing scope*"

```
public static void main(String[] args) {
    final String message = "Hello world";
    new Runnable() {
        @Override
        public void run() {
            System.out.println(message);
        }
    }.run();
}
```

But it will run without a problem with Java 8, since *message* variable will be marked as *final* implicitly:

```
public static void main(String[] args) {
    String message = "Hello world";
    new Runnable() {
        @Override
        public void run() {
            System.out.println(message);
        }
    }.run();
}
```