

# JAVA 7 & JDK7

**Author: Igor Blinov**  
**Education Manager**  
[ihar\\_blinou@epam.com](mailto:ihar_blinou@epam.com)

- **Language changes (Project Coin)**
- **Concurrency & Collection update**
- **Support for dynamically typed non-java languages**
- **Garbage First Collection**
- **Support Unicode 6.0.0**
- **Java XML: JAXB 2.2.3, JAXP 1.4.5, JAX-WS 2.2.4**
- **JDBC 4.1**
- **Add-in Swing & Java2D**
- **NIO.2**

## Language changes (Project Coin)

- Diamond
- Try-with-resources
- Multi-catch with more precise rethrow
- Enhanced literals & binary literals
- Strings in switch
- Safe varargs

## Diamond

```
List<Set<String>> list = new ArrayList<Set<String>>();  
List<String> strings = new ArrayList<String>();  
Map<Integer, String> map = new HashMap<Integer, String>();
```

---

```
List<Set<String>> list = new ArrayList<>();  
List<String> strings = new ArrayList<>();  
Map<Integer, String> map = new HashMap<>();
```

## Try-witch-resources

```
String filename = "path+filename";
BufferedWriter buf = null;
try {
    buf = new BufferedWriter(new FileWriter(filename));
    buf.write("some info");
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (buf != null) {
        try {
            buf.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

## Try-witch-resources

```
String filename = "path+filename";  
try (BufferedWriter buf = new BufferedWriter(new FileWriter(filename))) {  
    buf.write("some info");  
} catch (IOException e) {  
    e.printStackTrace();  
}
```

## Try-witch-resources

for any class that implements [java.lang.AutoCloseable](#) or/and [java.io.Closable](#)

```
public interface Closeable extends AutoCloseable {  
    void close() throws IOException;  
}
```

```
public interface AutoCloseable {  
    void close() throws Exception;  
}
```

## Try-witch-resources

```
String src = "srcfilename";
String dest = "destfilename";
    try (InputStream in = new FileInputStream(src);
        OutputStream out = new FileOutputStream(dest)) {
        // using in & out
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

## Multi-catch with more precise rethrow

```
String src = "srcfilename";
String dest = "destfilename";
try (InputStream in = new FileInputStream(src);
     OutputStream out = new FileOutputStream(dest)) {
    // using in & out
} catch (FileNotFoundException | IOException e) {
    e.printStackTrace();
}
```

## Multi-catch with more precise rethrow

```
try {  
    // some operations  
} catch(NumberFormatException e) {  
    e.printStackTrace();  
} catch(ClassNotFoundException e) {  
    e.printStackTrace();  
} catch(InstantiationException e) {  
    e.printStackTrace();  
}
```

## Multi-catch with more precise rethrow

```
try {  
    // some operations  
} catch(NumberFormatException | ClassNotFoundException |  
        InstantiationException e) {  
    e.printStackTrace();  
}
```

## Multi-catch with more precise rethrow

```
void reading() throws FileNotFoundException {  
    Scanner scan = null;  
    try {  
        FileReader fr = new FileReader("filename");  
        scan = new Scanner(fr);  
        System.out.println(scan.next());  
    } catch (Exception e) {  
        throw e; // compile error in Java 6  
    }  
}
```

## Enhanced integer literals & binary literals

```
long counter = 10000000000L;
```

---

```
long counter = 10_000_000_000L;
```

```
double d = 1_000.7;
```

**\_100\_000** or **10\_000\_** is Invalid!

```
int flag = 0b10;
```

## Strings in switch

```
public int defineLevel(String role) {  
    int level = 0;  
    switch (role) {  
        case "guest":  
            level = 1; break;  
        case "client":  
            level = 2; break;  
        case "moderator":  
            level = 3; break;  
        case "admin":  
            level = 4; break;  
        default: throw new IllegalArgumentException();  
    }  
    return level;  
}
```

## Safe varargs

### **@SuppressWarnings("varargs")**

```
static <T> void addToList (List<T> listArg, T... elements) {  
    for (T x : elements) {  
        listArg.add(x);  
    }  
}
```

### **@SuppressWarnings({"unchecked", "varargs"})**

```
static <T> void addToList (List<T> listArg, T... elements) {  
    for (T x : elements) {  
        listArg.add(x);  
    }  
}
```

## Concurrency & Collection update

- lightweight fork/join framework. Based on class **ForkJoinPool**
- flexible and reusable synchronization barriers: class **Phaser**
- transfer queues: interface `TransferQueue` & class **LinkedTransferQueue**
- thread-local pseudo-random number generators:  
class **ThreadLocalRandom**

## JDBC 4.1

```
public static void viewStudents(Connection con) throws SQLException {  
    String query = "SELECT * FROM student";  
    try (Statement stmt = con.createStatement()) {  
        ResultSet rs = stmt.executeQuery(query);  
        while (rs.next()) {  
            // operations  
        }  
    } catch (SQLException e) {  
        JDBCTutorialUtilities.printStackTrace(e);  
    }  
}
```

**Java.SE.00.7new**

**Игорь Блинов**

**PhD in Mathematics**

**Oracle Certified Java Instructor**

**blinov(a)gmail(dot)com**

**ihar\_blinou(a)epam(dot)com**

**skype: igor.n.blinov**

