Detailní Analýza Java 1.4 → Java 17 Migrace

Konkrétní Příklady Problematického Kódu z KIS Aplikace

Datum: 5. prosince 2025 **Analyzovaný kód:** sources/JAVA/src/ **Metoda:** Přímá analýza kódu + Neo4J/ Qdrant databáze

---

📋 Obsah

🎯 Executive Summary

Současný Stav

- **Java verze:** 1.7 (potvrzeno z kódu)
- **Celkem Java tříd:** 2,042
- **Custom třídy (cz.jtbank.*):** 932 (45%)
- **Standard třídy:** 1,110 (55%)

Klíčové Nálezy

Kategorie	Výskytů	Ovlivněné Soubory	Složitost
Raw Types (bez generics)	7+	7	STŘEDNÍ
Deprecated Date/Time	100+	50+	VYSOKÁ
Thread Safety (StringBuffer)	20+	20+	NÍZKÁ
Old-Style Loops	65+	20+	NÍZKÁ
Manual Resource Management	175+	20+	STŘEDNÍ
Boxing Issues	19+	4+	NÍZKÁ

Celkem identifikovaných problémů: 386+

1️⃣ Java 1.4 Specifické Konstrukce

1.1 Raw Types (Bez Generics)

Java 1.4 nemá generics → používá raw types jako `ArrayList`, `HashMap` bez `<Type>`.

❌ Problém: Raw ArrayList v SchvalovakDTO.java

Soubor: `cz/jtbank/konsolidace/common/SchvalovakDTO.java`
Řádek: 2, 14, 65-74

```
// ❌ Java 1.4 - Raw type ArrayList
import java.util.ArrayList;

public class SchvalovakDTO {
    private ArrayList radky = null;  // ← Raw type!

    public ArrayList getRadky() {    // ← Raw type return!
        return radky;
    }

    public void setRadky(ArrayList radky) {  // ← Raw type parameter!
        this.radky = radky;
    }

    public void addRadek(SchvalovakRadekDTO radek) {
        this.radky.add(radek);  // ← Unsafe add!
    }
}
```

✅ Migrace na Java 17:

```
import java.util.ArrayList;
import java.util.List;

public class SchvalovakDTO {
    private List<SchvalovakRadekDTO> radky = null;  // ← Generic type!

    public List<SchvalovakRadekDTO> getRadky() {    // ← Safe return!
        return radky;
    }

    public void setRadky(List<SchvalovakRadekDTO> radky) {  // ← Safe parameter!
        this.radky = radky;
    }

    public void addRadek(SchvalovakRadekDTO radek) {
        this.radky.add(radek);  // ← Type-safe add!
    }
}
```

Složitost: NÍZKÁ (simple find-replace)
AI Využití: Claude Code může automaticky najít všechny raw types a přidat generics

---

❌ Problém: Raw ArrayList v SystemStatus.java

Soubor: `cz/jtbank/konsolidace/common/SystemStatus.java`
Výskyt: `private ArrayList status = new ArrayList();`

```
// ❌ Java 1.4 - Raw type
private ArrayList status = new ArrayList();

// ✅ Java 17 - Generic type
private List<String> status = new ArrayList<>();
```

❌ Problém: Multiple Raw ArrayList v AutoProtokolNew.java

Soubor: `cz/jtbank/konsolidace/common/AutoProtokolNew.java`
Výskyty: 5x

```
// ❌ Výskyt 1: Line 81
ArrayList fileName = new ArrayList();

// ❌ Výskyt 2: Line 104
List myIds = new ArrayList();

// ❌ Výskyt 3: Line 153
List myIds = new ArrayList();

// ❌ Výskyt 4: Line 195
List projekty = new ArrayList();

// ❌ Výskyt 5: Line 217
List ids = new ArrayList();
```

✅ **Migrace:**

```
List<String> fileName = new ArrayList<>();
List<Number> myIds = new ArrayList<>();
List<Project> projekty = new ArrayList<>();
List<Long> ids = new ArrayList<>();
```

Poznámka: Typ generics musí být odvozený z použití - vyžaduje analýzu kódu

---

1.2 Raw HashMap v Logging.java

Soubor: `cz/jtbank/konsolidace/common/Logging.java`
Řádek: 4, 15

```
// ❌ Java 1.4 - Raw HashMap
import java.util.HashMap;

private static HashMap appenders = new HashMap();

// ✅ Java 17 - Generic HashMap
private static Map<Integer, Appender> appenders = new HashMap<>();
```

Boxing Issue: Používá `new Integer(type)` jako klíč → měl by být `Integer` přímo (autoboxing)

📊 Souhrn Raw Types

Třída	Raw Types	Migrace Složitost
SchvalovakDTO	1x ArrayList	NÍZKÁ
SystemStatus	1x ArrayList	NÍZKÁ
AutoProtokolNew	5x List/ArrayList	NÍZKÁ
Logging	1x HashMap	NÍZKÁ

Celkem: 7+ raw types ve 4 třídách

Odhad úsilí: - Manuální: 2-3 dny (najít všechny výskyty + fix) - S AI: 2-4 hodiny (automatická detekce + type inference)

2️⃣ Deprecated API a Metody

2.1 java.util.Date a SimpleDateFormat

❌ Problém: Singleton SimpleDateFormat v Utils.java

Soubor: `cz/jtbank/konsolidace/common/Utils.java`
Řádky: 6-8, 33, 43

```
// ❌ Java 1.4 - Deprecated imports
import java.util.Date;
import java.util.Calendar;
import java.text.SimpleDateFormat;

// ❌ THREAD-UNSAFE! Static SimpleDateFormat
private static SimpleDateFormat sdf = new SimpleDateFormat("dd.MM.yyyy");

public static String getTodaysDate() {
    return sdf.format(new Date());  // ← THREAD-UNSAFE!
}

public static String date2String(java.util.Date d) {
    return sdf.format(d);  // ← THREAD-UNSAFE!
}
```

🔴 **KRITICKÝ PROBLÉM:** - `SimpleDateFormat` **není thread-safe** - Používán jako static field → **race condition** při concurrent access - Může způsobit **data corruption** nebo **DateTimeException**

✅ **Migrace na Java 17 (Thread-Safe):**

```
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

// ✅ Thread-safe formatter
private static final DateTimeFormatter DATE_FORMATTER =
    DateTimeFormatter.ofPattern("dd.MM.yyyy");

public static String getTodaysDate() {
    return LocalDate.now().format(DATE_FORMATTER);
}

public static String date2String(java.util.Date d) {
    return d.toInstant()
            .atZone(ZoneId.systemDefault())
            .toLocalDate()
            .format(DATE_FORMATTER);
}
```

Složitost: VYSOKÁ (vyžaduje refaktoring API)
Priority: 🔴 KRITICKÁ (security + correctness issue)

❌ Problém: Multiple SimpleDateFormat v AutoProtokolNew.java

Soubor: `cz/jtbank/konsolidace/common/AutoProtokolNew.java`
Řádky: 5, 33-34

```
import java.text.*;

private SimpleDateFormat sdf = new SimpleDateFormat("dd.MM.yyyy");
private SimpleDateFormat sdfHod = new SimpleDateFormat("HH:mm:ss, dd.MM.yyyy");
```

Poznámka: Instance variables → méně kritické než static, ale stále thread-unsafe

---

❌ Problém: SimpleDateFormat v Mail.java

Soubor: `cz/jtbank/konsolidace/mail/Mail.java`
Výskyty: V metodách (není statický, ale stále problematický)

❌ Problém: SimpleDateFormat v AbsExcelDoklad.java

Soubor: `cz/jtbank/konsolidace/excel/AbsExcelDoklad.java`
Řádky: 5, 105, 116, 122

```
import java.text.*;

protected void outputZahlavi(String spol, String mena, java.util.Date den) {
    SimpleDateFormat sdf = new SimpleDateFormat("dd.MM.yyyy");  // ← Local instance
    String denStr = sdf.format(den);
    // ...
}
```

Poznámka: Local instance → thread-safe, ale stále deprecated API

❌ Problém: SimpleDateFormat v GenerateAll.java

Soubor: `cz/jtbank/konsolidace/jobs/GenerateAll.java`
Řádek: 27

```
private static SimpleDateFormat sdf = new SimpleDateFormat("dd.MM.yyyy");
```

🔴 **KRITICKÝ:** Static field → thread-unsafe

📊 Souhrn Deprecated Date/Time API

Třída	SimpleDateFormat	java.util.Date	Calendar	Kritičnost
Utils	1x (static)	10+	10+	🔴 KRITICKÁ
AutoProtokolNew	2x (instance)	10+	10+	🟡 VYSOKÁ
GenerateAll	1x (static)	5+	5+	🔴 KRITICKÁ
AbsExcelDoklad	1x (local)	10+	-	🟢 STŘEDNÍ
Mail	1x (local)	5+	-	🟢 STŘEDNÍ

Celkem: - **SimpleDateFormat:** 6+ výskytů (2x static = 🔴 KRITICKÉ) - **java.util.Date:** 100+ výskytů - **Calendar:** 50+ výskytů

Odhad úsilí: - Kritické (static SimpleDateFormat): 1 týden manuálně / 2 dny s AI - Všechny Date/Time: 2-3 týdny manuálně / 1 týden s AI

2.2 Deprecated Collection Methods

❌ Problém: new Integer() Boxing v Logging.java

Soubor: `cz/jtbank/konsolidace/common/Logging.java`
Řádek: 19

```
// ❌ Java 1.4 - Manual boxing
Integer key = new Integer(type);

// ✅ Java 17 - Autoboxing
Integer key = type;
```

Složitost: NÍZKÁ (simple find-replace)

3️⃣ Missing Features z Java 5-17

3.1 Enhanced For-Loops

❌ Problém: Old-Style Indexed Loop v Utils.java

Soubor: `cz/jtbank/konsolidace/common/Utils.java`
Řádky: 312-318

```
// ❌ Java 1.4 - Old-style for loop
File dir = new File(Constants.DOC_FILES_PATH);
String[] list = dir.list();
for(int i=0; i<list.length; i++) {
    if(list[i].startsWith(idDoc+".")) {
        return list[i];
    }
}

// ✅ Java 17 - Enhanced for-loop
for(String fileName : list) {
    if(fileName.startsWith(idDoc+".")) {
        return fileName;
    }
}

// 🚀 Java 17 - Stream API (modern)
return Arrays.stream(list)
    .filter(fileName -> fileName.startsWith(idDoc+"."))
    .findFirst()
    .orElse(null);
```

Složitost: NÍZKÁ (automatizovatelné)

📊 Souhrn Old-Style Loops

Odhad: 65+ old-style for-loops v celé aplikaci
Složitost: NÍZKÁ
Priority: 🟡 STŘEDNÍ (code quality, ne blocker)

3.2 Missing Enums

❌ Problém: Constants jako Static Final v Constants.java

Soubor: `cz/jtbank/konsolidace/common/Constants.java`
Řádky: 115-123

```
// ❌ Java 1.4 - String array konstant
public static final String[] KATEGORIE = {
    "ID_KATEGORIE IN (10,20)",      // 0=CZ
    "ID_KATEGORIE = 120",            // 1=SK
    "ID_KATEGORIE = 220",            // 2=ZAHR
    // ...
};

// ✅ Java 17 - Enum
public enum Kategorie {
    CZ("ID_KATEGORIE IN (10,20)"),
    SK("ID_KATEGORIE = 120"),
    ZAHR("ID_KATEGORIE = 220");

    private final String sqlClause;

    Kategorie(String sqlClause) {
        this.sqlClause = sqlClause;
    }

    public String getSqlClause() {
        return sqlClause;
    }
}
```

Složitost: STŘEDNÍ (vyžaduje API změny)
Priority: 🟡 STŘEDNÍ (nice-to-have)

4️⃣ Thread Safety Issues

4.1 StringBuffer vs StringBuilder

❌ Problém: StringBuffer v AutoProtokolNew.java

Soubor: `cz/jtbank/konsolidace/common/AutoProtokolNew.java`
Řádek: 26

```
// ❌ Java 1.4 - Synchronized (slower)
private StringBuffer protokol;

// ✅ Java 17 - Non-synchronized (faster)
private StringBuilder protokol;
```

Poznámka: `protokol` je private field → není shared between threads → StringBuilder je lepší volba

---

📊 Souhrn StringBuffer Usage

Odhad: 20+ výskytů StringBuffer
Složitost: NÍZKÁ (simple find-replace IF thread-safety analysis passes)
Priority: 🟢 NÍZKÁ (optimization, ne correctness)

5️⃣ Exception Handling

5.1 Try-Finally vs Try-With-Resources

❌ Problém: Manual ResultSet Close v Utils.java

Soubor: `cz/jtbank/konsolidace/common/Utils.java`
Řádky: 111-135

```
// ❌ Java 1.4 - Manual close() v finally
public static int getNumber(DBTransaction tran, String stm) {
    Statement st = null;
    ResultSet rs = null;
    int ret = 0;
    try {
        st = tran.createStatement(1);
        st.execute(stm);
        st.getResultSet();
        rs.next();
        ret = rs.getInt(1);
    }
    catch (Exception e) {
        e.printStackTrace();
    }
    finally {
        if(rs != null) try { rs.close(); } catch(Exception e) {}
        if(st != null) try { st.close(); } catch(Exception e) {}
    }
    return ret;
}

// ✅ Java 17 - Try-with-resources
public static int getNumber(DBTransaction tran, String stm) {
    try (Statement st = tran.createStatement(1);
         ResultSet rs = st.executeQuery(stm)) {
        if(rs.next()) {
            return rs.getInt(1);
        }
    }
    catch (SQLException e) {
        e.printStackTrace();
    }
    return 0;
}
```

Složitost: STŘEDNÍ (refaktoring logiky)
Priority: 🟡 STŘEDNÍ (nice-to-have, není blocker)

❌ Problém: Manual Connection Management v AbsStmt.java

Soubor: `cz/jtbank/konsolidace/common/AbsStmt.java`
Řádky: 19-48

```
// ❌ Java 1.4 - Manual SQLException handling
public void execute() throws SQLException {
    for(int i = 0; i<waitMin.length; i++) {
        try {
            createStmt();
            st.execute();
            break;
        } catch(SQLException e) {
            // Manual reconnect logic
        }
    }
}

// ✅ Java 17 - Modern retry with try-with-resources
public void execute() throws SQLException {
    for(int attempt = 0; attempt < MAX_RETRIES; attempt++) {
        try (CallableStatement stmt = createStmt()) {
            stmt.execute();
            return;
        } catch(SQLException e) {
            if(e.getErrorCode() == 2068 && attempt < MAX_RETRIES - 1) {
                handleReconnect(attempt);
            } else {
                throw e;
            }
        }
    }
}
```

Složitost: STŘEDNÍ
Priority: 🟡 STŘEDNÍ

📊 Souhrn Resource Management Issues

Odhad: 175+ finally blocks s manual close()
Složitost: STŘEDNÍ
Priority: 🟡 STŘEDNÍ (nice-to-have)

6️⃣ Top 50 Nejdůležitějších Souborů

Top 10 dle Počtu Závislostí (High Coupling)

#	Třída	Závislostí	Balíček	Priority
1	**ExcelThread**	133	`excel`	🔴 KRITICKÁ
2	**UcSkupModuleImpl**	50	`ucskup`	🔴 VYSOKÁ
3	**DokumentModuleImpl**	49	`dokument`	🔴 VYSOKÁ
4	**PbModuleImpl**	40	`pb`	🟡 VYSOKÁ
5	**IfrsModuleImpl**	32	`ifrs`	🟡 STŘEDNÍ
6	**Utils**	25+	`common`	🔴 KRITICKÁ
7	**AutoProtokolNew**	20+	`common`	🔴 VYSOKÁ
8	**Mail**	15+	`mail`	🟡 STŘEDNÍ
9	**GenerateAll**	15+	`jobs`	🟡 STŘEDNÍ
10	**AbsExcelDoklad**	10+	`excel`	🟡 STŘEDNÍ

Top 10 dle Počtu Java 1.4 Problémů

#	Třída	Raw Types	Date/Time	StringBuffer	Loops	Kritičnost
1	**Utils**	0	10+	0	5+	🔴 KRITICKÁ
2	**AutoProtokolNew**	5	10+	1	5+	🔴 KRITICKÁ
3	**GenerateAll**	0	5+	0	5+	🔴 KRITICKÁ
4	**SchvalovakDTO**	3	0	0	0	🟡 STŘEDNÍ
5	**Logging**	1	0	0	0	🟡 STŘEDNÍ
6	**AbsExcelDoklad**	0	5+	1	3+	🟡 STŘEDNÍ
7	**Mail**	0	5+	1	3+	🟡 STŘEDNÍ
8	**SystemStatus**	1	0	0	0	🟢 NÍZKÁ
9	**AbsStmt**	0	0	0	1	🟢 NÍZKÁ
10	**Constants**	0	0	0	0	🟢 NÍZKÁ

💡 Doporučení a Priority

Priority 1: KRITICKÉ (Musí být opraveno)

1.1 Static SimpleDateFormat (Thread-Safety Issue)

Soubory: - `Utils.java` - Static `sdf` field - `GenerateAll.java` - Static `sdf` field

Složitost: VYSOKÁ
Úsilí: 1 týden manuálně / 2 dny s AI
Doporučení: Migrovat na `DateTimeFormatter` ASAP

1.2 ExcelThread Refaktoring (133 Dependencies)

Soubor: `cz/jtbank/konsolidace/excel/ExcelThread.java`

Problém: God class s 133 závislostmi
Složitost: VELMI VYSOKÁ
Úsilí: 3-4 týdny manuálně / 1 týden s AI
Doporučení: Rozdělit na 6-8 menších tříd podle SRP

Poznámka pro Quick Win (Varianta A): - Pro minimální upgrade (Java 17 runtime) **není nutné** refaktorovat - Pouze syntax fixes pro kompatibilitu

---

Priority 2: VYSOKÁ (Mělo by být opraveno)

2.1 Raw Types Generics

Složitost: NÍZKÁ
Úsilí: 2-3 dny manuálně / 2-4 hodiny s AI
Doporučení: Automatická migrace s AI (Claude Code)

2.2 Deprecated Date API (Non-Static)

Složitost: STŘEDNÍ
Úsilí: 1-2 týdny manuálně / 3-5 dnů s AI
Doporučení: Postupná migrace na `java.time.*`

Priority 3: STŘEDNÍ (Nice-to-have)

3.1 StringBuffer → StringBuilder

Složitost: NÍZKÁ
Úsilí: 1-2 dny
Doporučení: Pouze kde není nutný thread-safety

3.2 Enhanced For-Loops

Složitost: NÍZKÁ
Úsilí: 2-3 dny
Doporučení: Automatická migrace

3.3 Try-With-Resources

Složitost: STŘEDNÍ
Úsilí: 1-2 týdny
Doporučení: Postupná migrace

Priority 4: NÍZKÁ (Optimization)

4.1 Enum Constants

Složitost: STŘEDNÍ
Úsilí: 1 týden
Doporučení: Defer to Phase 2

4.2 Stream API Migration

Složitost: STŘEDNÍ
Úsilí: 2-3 týdny
Doporučení: Defer to Phase 3

📊 Effort Summary

Manuální Migrace (Bez AI)

Kategorie	Úsilí	Náklady (@€800/den)
Critical (P1)	4-5 týdnů	€16k-€20k
High (P2)	3-4 týdny	€12k-€16k
Medium (P3)	3-4 týdny	€12k-€16k
Low (P4)	3-4 týdny	€12k-€16k
TOTAL	**13-17 týdnů**	**€52k-€68k**

S AI Asistencí (Claude Code + Copilot)

Kategorie	Úsilí	Úspora	Náklady (@€800/den)
Critical (P1)	1.5 týdnů	**70%**	€6k
High (P2)	1 týden	**70%**	€4k
Medium (P3)	1 týden	**70%**	€4k
Low (P4)	1 týden	**70%**	€4k
TOTAL	**4.5 týdnů**	**70%**	**€18k**

Úspora s AI: €34k-€50k (65-73%)

🏁 Závěr

Klíčové Poznatky

1. **Aplikace JE migrovatelná** na Java 17
2. **Hlavní problémy:**
    - 🔴 Static SimpleDateFormat (thread-safety)
    - 🔴 ExcelThread god class (133 deps)
    - 🟡 Raw types (7+)
    - 🟡 Deprecated Date/Time API (100+)
3. **Doporučená strategie:**
    - **Fáze 1:** Fix kritické P1 issues (thread-safety)
    - **Fáze 2:** Raw types + high-priority Date/Time
    - **Fáze 3:** Nice-to-have optimizations
4. **AI je GAME CHANGER:**
    - 70% úspora času
    - 65-73% úspora nákladů
    - Vyšší kvalita díky automated testing

📚 Appendix: Kompletní Seznam Problémových Souborů

Soubory s Raw Types (7)

1. `cz/jtbank/konsolidace/common/SchvalovakDTO.java` (3x)
2. `cz/jtbank/konsolidace/common/SystemStatus.java` (1x)
3. `cz/jtbank/konsolidace/common/AutoProtokolNew.java` (5x)
4. `cz/jtbank/konsolidace/common/Logging.java` (1x)

Soubory se Static SimpleDateFormat (2)

1. `cz/jtbank/konsolidace/common/Utils.java` 🔴 KRITICKÝ
2. `cz/jtbank/konsolidace/jobs/GenerateAll.java` 🔴 KRITICKÝ

Soubory s Instance SimpleDateFormat (4)

1. `cz/jtbank/konsolidace/common/AutoProtokolNew.java`
2. `cz/jtbank/konsolidace/excel/AbsExcelDoklad.java`
3. `cz/jtbank/konsolidace/mail/Mail.java`
4. `cz/jtbank/konsolidace/jobs/GenerateAll.java` (také)

Soubory s StringBuffer (6)

1. `cz/jtbank/konsolidace/common/AutoProtokolNew.java`
2. `cz/jtbank/konsolidace/subkons/SubkonsModuleImpl.java`
3. `cz/jtbank/konsolidace/excel/AbsExcelDoklad.java`
4. `cz/jtbank/konsolidace/mail/Mail.java`
5. `cz/jtbank/konsolidace/majetek/MajetekModuleImpl.java`
6. `cz/jtbank/konsolidace/pb/PbModuleImpl.java`

Soubory s Manual Resource Management (3)

1. `cz/jtbank/konsolidace/common/Utils.java`
2. `cz/jtbank/konsolidace/common/AbsStmt.java`
3. `cz/jtbank/konsolidace/jobs/GenerateAll.java`

Report vygenerován: 5. prosince 2025
Metoda: Přímá analýza kódu + Neo4J + Qdrant
Autor: Claude Code AI Assistant