# Erros Mais Comuns em Java (Versões LTS) e Como Corrigir

Como profissional que atua ativamente em projetos baseados em Java desde a versão 8 até a 17, reuni neste artigo um guia completo e prático com os erros mais comuns que programadores enfrentam ao trabalhar com Java nas versões LTS.

Meu objetivo é ajudar tanto iniciantes quanto desenvolvedores experientes a compreenderem por que esses erros ocorrem e como evitá-los com boas práticas de código, uso de ferramentas e análise criteriosa do fluxo da aplicação.

Cada erro listado aqui vem acompanhado de uma explicação clara sobre a causa, exemplos práticos e dicas diretas para resolvê-lo, tornando este artigo uma referência tanto para estudo quanto para consulta no dia a dia de desenvolvimento.

## Erros de Compilação (Compile-time Errors)

Esses erros impedem o código de ser compilado e são os primeiros que você encontra ao programar em Java.

### cannot find symbol

**Causa:** Nome incorreto ou símbolo não declarado.
**Exemplo:**

*System.ou.println("Olá"); // erro em "ou"*

**Solução:** Corrigir para System.out.println.

### incompatible types

**Causa:** Tipos de dados incompatíveis.

*int numero = "dez"; // erro*

**Solução:** Usar Integer.parseInt("10").

### ';' expected

**Causa:** Faltou ponto e vírgula.

*System.out.println("Oi") // erro*

**Solução:** Adicionar ;.

### class, interface, or enum expected

**Causa:** Código fora de uma classe.
**Solução:**

```
public class Main {
   public static void main(String[] args) {
      // código aqui
   }
}
```

### illegal start of expression

**Causa:** Sintaxe incorreta, como chave fora do lugar.
**Solução:** Corrigir estrutura do código.

### missing return statement

**Causa:** Método com retorno obrigatório não retorna nada.

**Exemplo:**

```
public int somar(int a, int b) {
   int total = a + b;
} // erro
```

**Solução:** Adicionar return total;.

### variable might not have been initialized

**Causa:** Uso de variável local não inicializada.
**Solução:**

```
int idade = 0;
System.out.println(idade);
```

### non-static variable cannot be referenced from a static context

**Causa:** Uso de variável de instância dentro do main.
**Solução:**

```
Main obj = new Main();
System.out.println(obj.nome);
```

### package does not exist

**Causa:** Importação inválida.
**Solução:** Verificar nome do pacote ou dependência.

### unreachable statement

**Causa:** Código após return, throw, break.
**Solução:** Remover ou reestruturar o código.

## Erros em Tempo de Execução (Runtime Errors)

Aparecem somente quando o programa é executado.

### NullPointerException

**Causa:** Acesso a método ou campo de objeto nulo.
**Solução:** Verificar se o objeto foi instanciado.

### ArrayIndexOutOfBoundsException

**Causa:** Índice inválido em array.
**Solução:** Verificar com array.length.

### ClassCastException

**Causa:** Cast inválido.
**Solução:** Usar instanceof.

### NumberFormatException

**Causa:** Conversão inválida de String para número.
**Solução:** Validar entrada com regex ou try-catch.

### IllegalArgumentException

**Causa:** Argumento inadequado em método.
**Solução:** Validar argumentos.

### IllegalStateException

**Causa:** Objeto em estado inválido.
**Solução:** Garantir estado adequado.

### ArithmeticException

**Causa:** Divisão por zero.
**Solução:** Verificar divisor antes.

### ConcurrentModificationException

**Causa:** Modificação da lista durante iteração.
**Solução:** Usar Iterator com .remove() ou CopyOnWriteArrayList.

### StackOverflowError

**Causa:** Recursão infinita.
**Solução:** Definir condição de parada.

### OutOfMemoryError

**Causa:** Heap cheia.
**Solução:** Liberar objetos ou aumentar -Xmx.

## Erros com Threads e Concorrência

- **InterruptedException:** Tratar interrupções corretamente.
- **IllegalMonitorStateException:** Usar wait/notify apenas dentro de synchronized.
- **ReentrantLock not released:** Sempre usar try-finally.
- **Deadlock:** Evitar bloqueios circulares.
- **Race Condition:** Sincronizar acesso com synchronized, Atomic*, Locks.

## Erros com Streams e Arquivos

- **FileNotFoundException:** Arquivo não localizado.
- **IOException:** Erro genérico de entrada/saída.
- **UncheckedIOException:** Erro propagado não verificado.
- **MalformedURLException:** URL malformada.
- **URISyntaxException:** URI inválida.

## Erros com Reflection e ClassLoader

- **ClassNotFoundException:** Classe não encontrada em tempo de execução.
- **NoClassDefFoundError:** Classe compilada ausente no runtime.
- **InvocationTargetException:** Erro ao chamar método via reflection.
- **IllegalAccessException:** Acesso negado a membros privados.
- **InstantiationException:** Tentativa de instanciar interface ou classe abstrata.

## Erros com Coleções e Generics

- **ClassCastException com generics:** Usar generics corretamente.
- **UnsupportedOperationException:** Tentativa de modificar lista imutável.
- **EmptyStackException:** Acesso a pilha vazia.
- **ConcurrentModificationException:** Iteração com modificação simultânea.
- **MissingResourceException:** Arquivo .properties ausente ou mal localizado.

# Dica do EducaCiência para Iniciantes

Foque primeiro nestes erros:

- NullPointerException
- cannot find symbol
- ';' expected
- ArrayIndexOutOfBoundsException
- missing return statement
- incompatible types
- non-static variable cannot be referenced from a static context

Entender esses erros é essencial para dominar o Java.

Evitar erros em Java exige disciplina e organização. Algumas práticas essenciais:

- **Use IDEs modernas** (IntelliJ, Eclipse): alertam sobre erros em tempo real.
- **Documente com JavaDoc:** facilita entendimento do código.
- **Crie testes unitários (JUnit):** garante que métodos funcionem corretamente.
- **Evite duplicação:** siga o princípio DRY (Don't Repeat Yourself).
- **Trate exceções com responsabilidade:** evite blocos catch vazios.
- **Use análise estática:** SonarQube, PMD, Checkstyle ajudam a detectar problemas.
- **Refatore regularmente:** melhora a legibilidade e manutenção.

Compreender esses erros e adotar boas práticas torna você um desenvolvedor Java mais eficiente e confiável.

# Referências Bibliográficas


- Oracle Java Documentation: https://docs.oracle.com/en/java/javase/17/
- *Effective Java*, Joshua Bloch, 3ª edição, Addison-Wesley, 2018
- *Java: The Complete Reference*, Herbert Schildt, 11ª edição, McGraw-Hill, 2019
- Java Language Specification: https://docs.oracle.com/javase/specs/
- Baeldung Java Tutorials: https://www.baeldung.com/
- GeeksforGeeks Java Programming: https://www.geeksforgeeks.org/java/


EducaCiência FastCode para a comunidade