Introduction to Java

Java is a versatile, object-oriented programming language widely used for building a variety of applications, from enterprise software to mobile apps and games. It is known for its portability, security, and robust performance, making it a popular choice among developers worldwide.

by Laxman Deora

History and Evolution of Java

1

1991 - Oak

Java's origins trace back to the Oak programming language, developed by James Gosling and his team at Sun Microsystems.

2

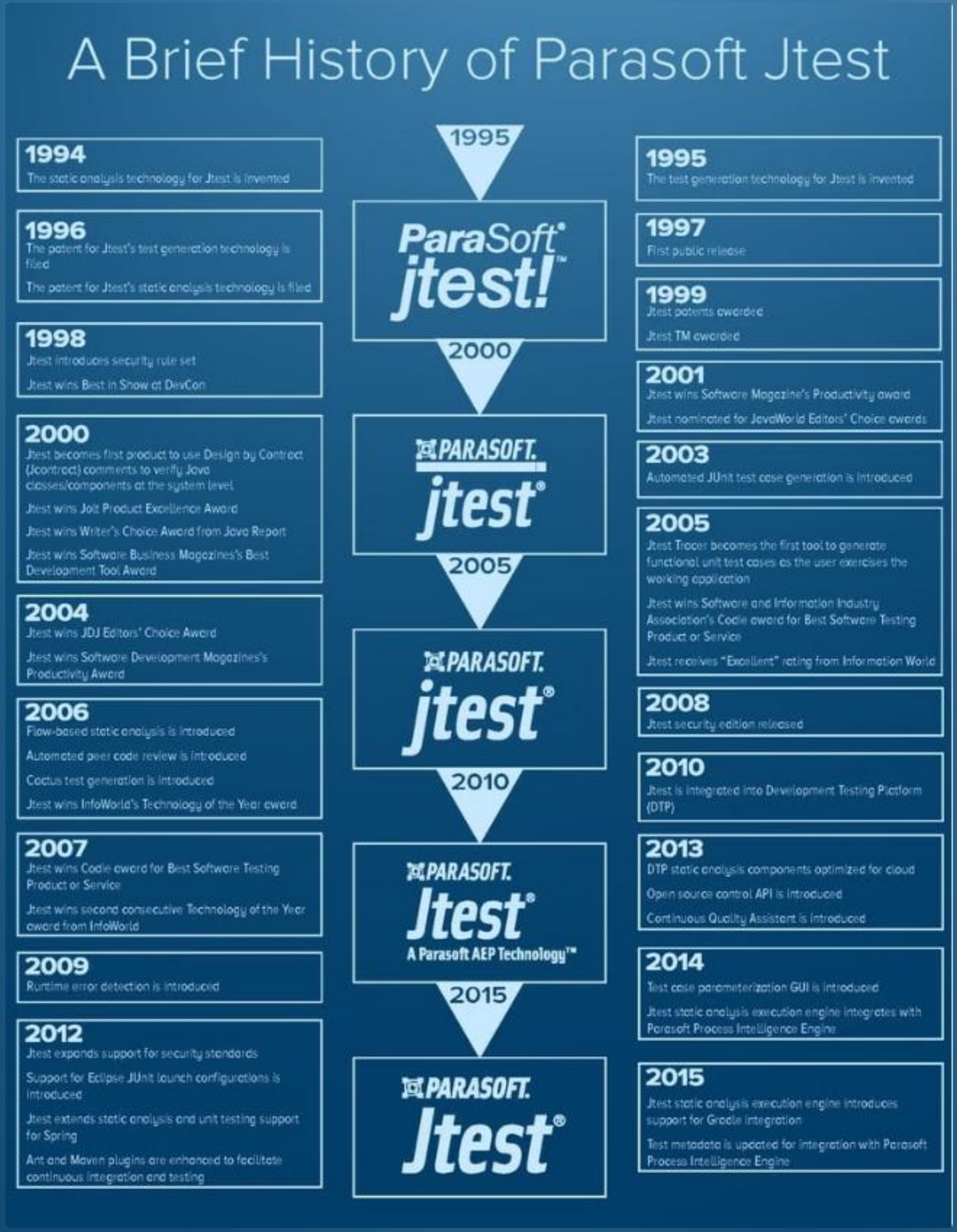
1995 - Java

Oak was renamed to Java and released publicly, becoming a popular choice for web applications and applets.

3

2006 - Java SE 6

The release of Java SE 6 brought significant improvements in performance, security, and developer productivity.


1995
2000
2005
2010
2015

A Brief History of Parasoft Jtest

1994
The static analysis technology for Jtest is invented

1996
The patent for Jtest's test generation technology is filed
The patent for Jtest's static analysis technology is filed

1998
Jtest introduces security rule set
Jtest wins Best in Show at DevCon

2000
Jtest becomes first product to use Design by Contract (Jcontract) comments to verify Java classes/components at the system level
Jtest wins Jolt Product Excellence Award
Jtest wins Writer's Choice Award from Java Report
Jtest wins Software Business Magazines's Best Development Tool Award

2004
Jtest wins JDJ Editors' Choice Award
Jtest wins Software Development Magazines's Productivity Award

2006
Flow-based static analysis is introduced
Automated peer code review is introduced
Cactus test generation is introduced
Jtest wins InfoWorld's Technology of the Year award

2007
Jtest wins Codie award for Best Software Testing Product or Service
Jtest wins second consecutive Technology of the Year award from InfoWorld

2009
Runtime error detection is introduced

2012
Jtest expands support for security standards
Support for Eclipse JUnit launch configurations is introduced
Jtest extends static analysis and unit testing support for Spring
Ant and Maven plugins are enhanced to facilitate continuous integration and testing

1995
The test generation technology for Jtest is invented

1997
First public release

1999
Jtest patents awarded
Jtest TM awarded

2001
Jtest wins Software Magazine's Productivity award
Jtest nominated for JavaWorld Editors' Choice awards

2003
Automated JUnit test case generation is introduced

2005
Jtest Tracer becomes the first tool to generate functional unit test cases as the user exercises the working application
Jtest wins Software and Information Industry Association's Codie award for Best Software Testing Product or Service
Jtest receives "Excellent" rating from Information World

2008
Jtest security edition released

2010
Jtest is integrated into Development Testing Platform (DTP)

2013
DTP static analysis components optimized for cloud
Open source control API is introduced
Continuous Quality Assistant is introduced

2014
Test case parameterization GUI is introduced
Jtest static analysis execution engine integrates with Parasoft Process Intelligence Engine

2015
Jtest static analysis execution engine introduces support for Gradle integration
Test metadata is updated for integration with Parasoft Process Intelligence Engine

Java as an Object-Oriented Programming Language

Encapsulation

Java enforces data encapsulation, where data and the methods that operate on that data are bundled together within a class.

Inheritance

Java supports inheritance, allowing classes to inherit properties and methods from parent classes, promoting code reuse.

Polymorphism

Java enables polymorphism, where objects of different classes can be treated as objects of a common superclass.

Key Features of Java

1. **Platform Independence**

   Java's "Write Once, Run Anywhere" (WORA) principle allows programs to run on various platforms without the need for recompilation.

2. **Robust Exception Handling**

   Java's exception handling mechanism provides a structured way to handle and recover from runtime errors.

3. **Automatic Memory Management**

   Java's Garbage Collector automatically reclaims memory occupied by objects that are no longer in use.

4. **Rich API**

   Java's extensive and well-documented standard library provides a wide range of pre-built functionality for developers.

Java Virtual Machine (JVM)

Platform Independence

The JVM allows Java programs to run on different operating systems without the need for recompilation.

Automatic Memory Management

The JVM's Garbage Collector automatically frees up memory occupied by objects that are no longer in use.

Bytecode Execution

The JVM executes Java's compiled bytecode, providing a consistent runtime environment for Java applications.

Security

The JVM's sandbox model helps ensure the security and stability of Java programs by isolating them from the underlying system.

Java Data Types and Variables

Primitive Data Types

Java offers a range of primitive data types, including integers, floating-point numbers, characters, and booleans.

Reference Data Types

Java also supports reference data types, such as strings, arrays, and user-defined classes, which can store complex data structures.

Variables

Variables in Java are used to store data, and can be declared using a variety of modifiers to control their scope and accessibility.

Java Control Structures and Conditional Statements

1

If-Else

The if-else statement allows you to execute different code blocks based on a boolean condition.

2

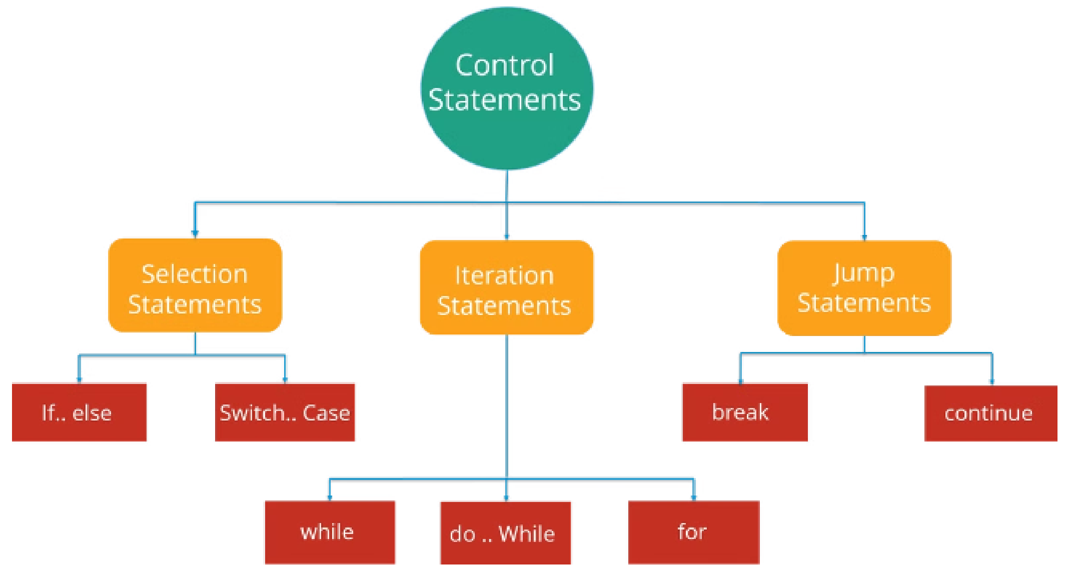
Switch

The switch statement provides a more concise way to handle multiple conditions and branch execution accordingly.

3

Loops

Java offers various loop constructs, such as for, while, and do-while, to repeatedly execute a block of code.



Java Methods and Functions

Methods

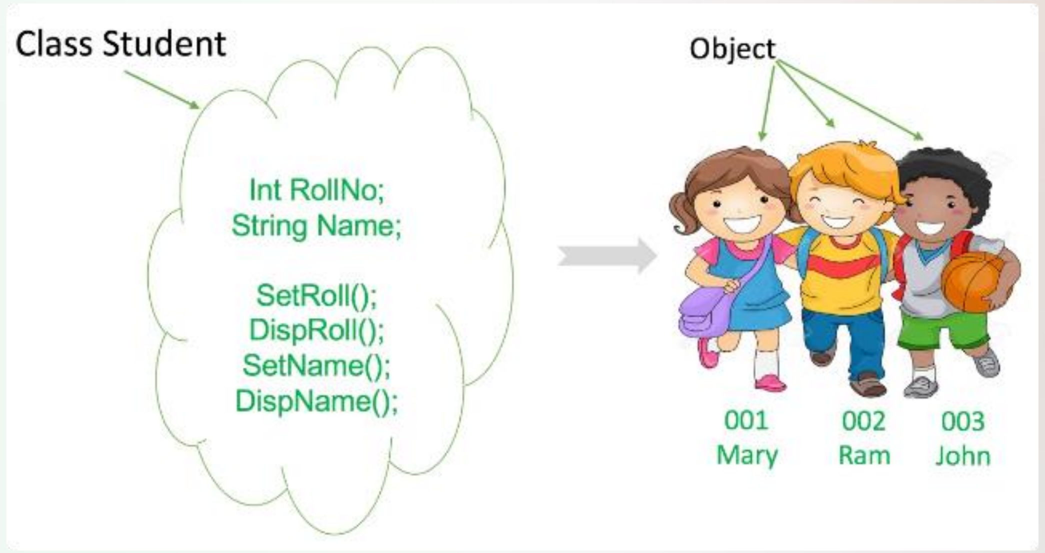
Methods are named blocks of code that perform specific tasks and can accept parameters and return values.

Functions

Functions are self-contained units of code that can be called with arguments and return a value.

Overloading

Java allows you to define multiple methods with the same name but different parameter lists, known as method overloading.



Java Classes and Objects

1

Class

A class is a blueprint or template that defines the properties and behaviors of an object.

2

Object

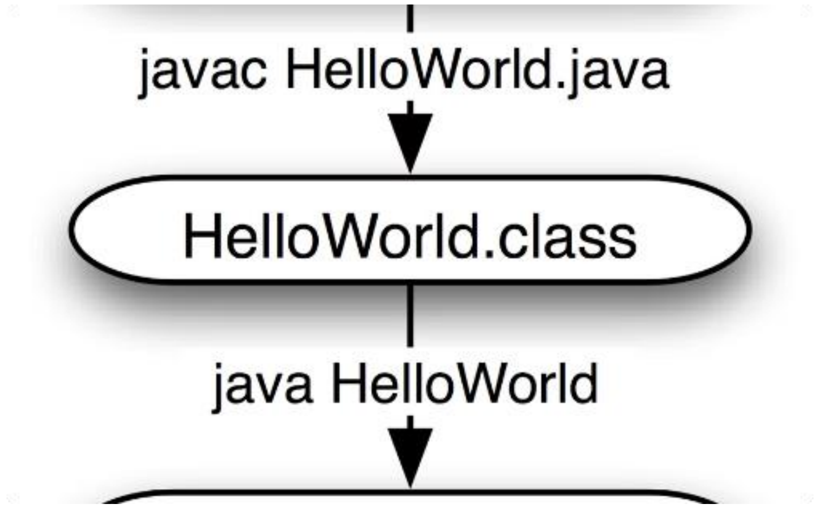
An object is an instance of a class, created using the new keyword, and can access the class's methods and properties.

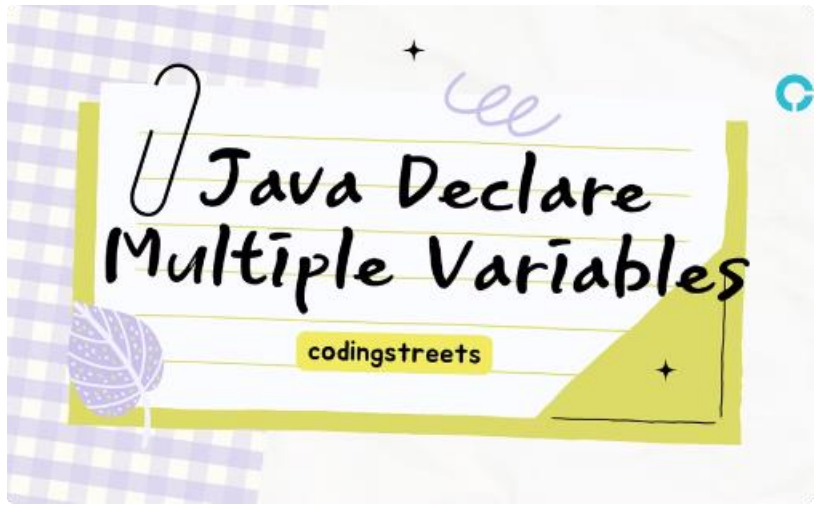
3

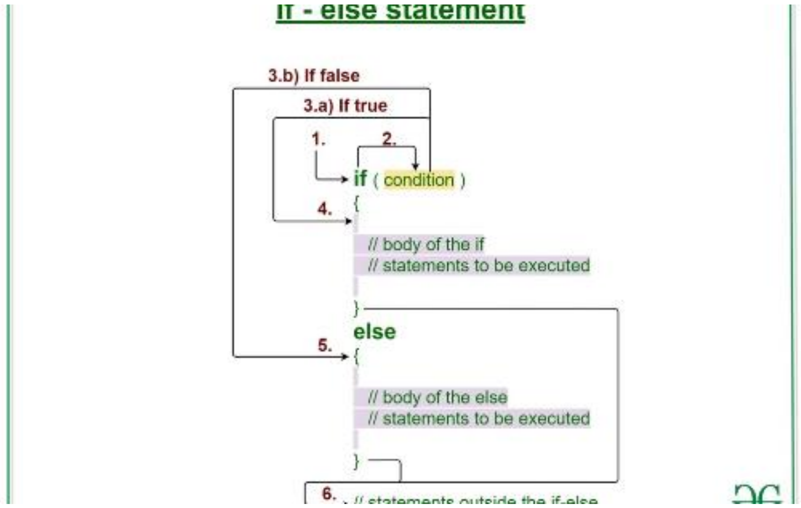
Inheritance

Classes can inherit properties and methods from a parent class, allowing for code reuse and polymorphism.

Simple Java Programming Examples







Hello World

A classic program that prints "Hello, World!" to the console, demonstrating the basic structure of a Java program.

Variable Declaration

An example of how to declare and initialize variables of different data types in Java.

Conditional Statements

An example of using the if-else statement to make decisions based on a condition in a Java program.