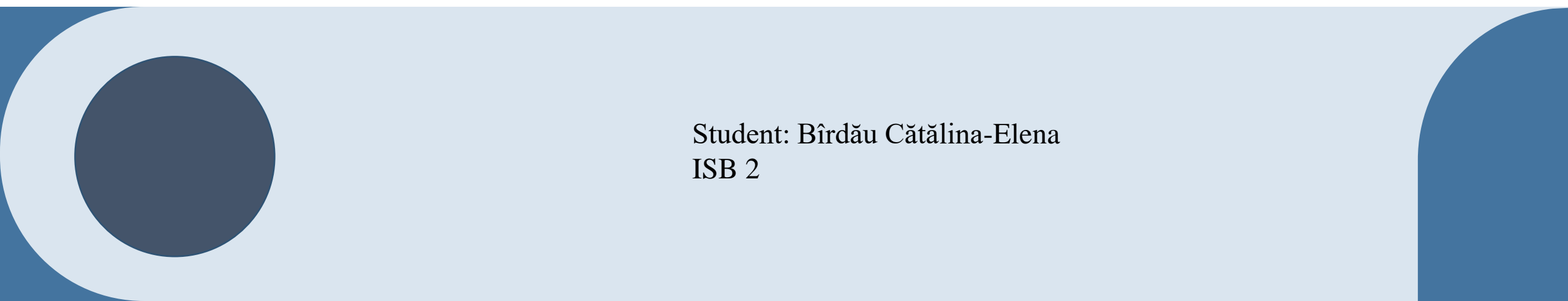




Self-Balancing Trees Simulation and Performance Evaluation

**A Comparative Study of AVL and Red-Black Trees in
Java and C++**



Student: Bîrdău Cătălina-Elena
ISB 2

Agenda

- ❖ Introduction to the Project
- ❖ Tools and Technologies Used
- ❖ Self-Balancing Trees Overview
- ❖ Key Operations in AVL and Red-Black Trees
- ❖ Performance Measurement Techniques
- ❖ Experimental Results - Simulation 1
- ❖ Key Observations (Simulation 1)
- ❖ Comparative Analysis - Insertion, Search, and Deletion
- ❖ Conclusion

Introduction to the Project

Goal: Develop an application to simulate and measure performance of **AVL Trees** and **Red-Black Trees**.

Key Operations Evaluated:

- **Insertion**
- **Deletion**
- **Search**

Comparison of Performance:

Between **Java** and **C++** implementations.

- Focus on execution time and memory usage.

Tools and Technologies Used

Programming Languages

- Java: Using the Java Development Kit (JDK 17) and standard libraries for performance analysis.
- C++: Using modern C++ standards (C++14) and libraries such as <chrono> for performance measurement.

IDEs

- Java: Eclipse 2022.
- C++: Visual Studio 2022.



Self-Balancing Trees Overview (1)

AVL Tree

- Self-balancing binary search tree.
- **Balance Factor:** Difference between left and right subtree heights must be -1, 0, or 1.
- **Rotation Mechanisms:**
 - Left Rotation
 - Right Rotation
 - **Double Rotations** when balance is violated.



Self-Balancing Trees Overview (2)

Red-Black Tree

- Self-balancing binary search tree.
- **Key Properties:**
 - Each node is **red** or **black**.
 - The **root** is always **black**.
 - **No two red nodes** can be adjacent.
 - **Every path** from a node to its leaves contains the same number of black nodes.



Key Operations in AVL and Red-Black Trees

Insertion:

- Both trees insert nodes similarly to a standard BST.
- After insertion, the tree is **rebalanced** using rotations (AVL) or recoloring (Red-Black).

Deletion:

- Standard BST deletion, followed by **rebalancing** or **recoloring**.

Search:

- Standard BST traversal based on node key comparison.



Performance Measurement Techniques

Time Measurement:

- **Java:** Used `System.nanoTime()` to measure operation times in nanoseconds.
- **C++:** Used `<chrono>` library for precise time measurement.

Memory Measurement:

- **Java:** Used `Runtime.getRuntime()` to track memory usage before and after operations.
- **C++:** Used `GetProcessMemoryInfo` on Windows to monitor memory usage.



Experimental Results - Simulation 1

Input Configuration:

- **numElements** = 10,000,000
- **bound** = 1,000,000,000
- **searchKey** = 133857562
- **deleteKey** = 298200287

Language	Tree Type	Insertion Time (μs)	Search Time (μs)	Delete Time (μs)	Memory Before Insert (KB)	Memory After Insert (KB)	Memory Change (KB)
Java	AVL Tree	9,257,248.10	4.2	16.3	795,650	940,032	144,382
Java	Red-Black Tree	8,650,777.40	6.1	9.001	1,209,550	1,468,620	259,070
C++	AVL Tree	24,382,687.60	0.2	4.7	52,372	986,632	934,260
C++	Red-Black Tree	11,540,076.70	0.3	1.8	52,376	1,148,236	1,095,860



Key Observations (Simulation 1)

- **Insertion:** Java AVL insertion is faster than C++ AVL.
- **Search:** C++ trees have superior search performance (Red-Black slower than AVL).
- **Deletion:** C++ Red-Black trees outperform Java in deletion speed.
- **Memory Usage:** Java uses more memory but with smaller increases during insertions.



Comparative Analysis - Insertion, Search, and Deletion

Insertion:

- Java performs better for large datasets due to **optimized memory management** and **garbage collection**.

Search:

- C++ has a clear advantage in search time due to **low-level memory control** and **cache optimization**.

Deletion:

- C++ Red-Black trees show faster deletion due to more **efficient tree restructuring** and lower memory overhead.



Conclusion

- **Java**: Ideal for **insertion-heavy applications** with automatic memory management.
- **C++**: Perfect for **performance-critical applications** requiring fast search and deletion with **minimal memory overhead**.
- Java is easier to use, while C++ excels in performance.



Thank you