Right Triangles Program 3 - Test Plan

Program Overview

Counts right triangles from a set of 2D points using direction vectors.
Uses PointStore abstraction to support text and binary input formats.

Files:
- Triangles.java: Single-process baseline
- ProcessTriangles.java: Multi-process coordinator (uses pipes for IPC)
- ThreadTriangles.java: Multi-thread version (uses shared memory)
- SingleProcessTriangleCounter.java: Child process worker
- PointStore.java: Storage interface
- TextPointStore.java: Text-encoded point reader
- BinPointStore.java: Binary-encoded point reader (memory-mapped I/O)
- TrianglesUtils.java: Shared algorithm (O(n^2))

Exit Codes: 1 = usage error, 2 = file error, 3 = format error

Test Cases

Functional Tests (Text Encoding):

Test File	Points	Expected	Description
single_right_triangle.txt	3	1	Minimum valid case
square_points.txt	4	4	Square corners
test_spec_list.txt	5	4	Professor's test
test_long_list.txt	5000	32909	Large dataset

Functional Tests (Binary Encoding):

Test File	Points	Expected	Description
single_right_triangle.dat	3	1	Binary version of 3-point case

Running Tests

Compile:
```
javac com/tryright/*.java
```

Single-process:
```
java com.tryright.Triangles test/<testfile>
```

Multi-process:
```
java com.tryright.ProcessTriangles test/<testfile> <num_processes>
```

Multi-thread:
```
java com.tryright.ThreadTriangles test/<testfile> <num_threads>
```

Example:
```
  java com.tryright.ProcessTriangles test/test_long_list.txt 8
  java com.tryright.ThreadTriangles test/test_long_list.txt 8
  java com.tryright.Triangles test/single_right_triangle.dat
```

Performance Results - test_long_list.txt (5000 points)

Tested on Windows (8 logical cores)

Workers	Process (s)	Thread (s)	Thread Speedup
1	8.59	8.38	1.02x
2	4.68	4.53	1.03x
4	3.10	2.95	1.05x
8	3.22	2.41	1.34x
16	3.72	2.75	1.35x
32	3.97	2.76	1.44x

Analysis

Threads outperform processes at higher worker counts due to:
1. No JVM startup overhead per worker
2. No pipe overhead (processes still pay per-child startup and I/O)
3. Direct shared memory access vs IPC

At low worker counts (1-2), overhead differences are minimal.
At higher counts (8+), threads show 30-45% improvement.

Processes degrade at high worker counts (e.g. 32 processes = 13.3s vs 4.5s
for threads on 5000 points) due to cumulative JVM startup and pipe overhead.
Threads remain flat because shared memory has no per-worker IPC cost.

Optimal configuration: 8 threads (matches CPU logical cores, best performance)

Binary Format Verification

Binary input files (.dat) are parsed using BinPointStore and MMIO. Results are
verified by comparing .dat output to the equivalent .txt tests (single_right_triangle,
test_spec_list, test_long_list).

Binary vs Text (8 Threads)

Results from ThreadTriangles with 8 threads on Windows:

Test File Pair	Expected	TXT Time (ms)	DAT Time (ms)
test_spec_list (.txt/.dat)	4	571.55	569.57
test_long_list (.txt/.dat)	32909	2868.18	2792.53