Part 1

Design Alternative	Pros	Cons
Design 1	Adaptable to both types of coordinates; stores them with 100% accuracy until converted	Needs extra storage for the flag; conditional logic can increase code complexity and slow down operations; inefficiencies when converting to the opposite coordinate system
Design 2	Requires only one variable pair to store; faster for polar coordinate returns	Will lose accuracy when storing cartesian coordinates; less efficient when cartesian coordinates are needed
Design 3	Requires only one variable pair to store; faster for cartesian coordinate returns	Will lose accuracy when storing polar coordinates; less efficient when polar coordinates are needed
Design 4	No conversion loss; faster retrieval for both coordinate types	Takes up more storage; potentially more memory-intensive
Design 5	Allows for specialized behavior depending on the subclass, leading to faster storage and no conversion loss	May require dynamic type checking, slowing down conversions; adds complexity due to the inheritance hierarchy

The tests were executed through the test application, compiled and run within the design2 and design5 directories (design2 directory also included a symbolic link to PointCP3). I switched these directories were in alternating order to ensure fair testing conditions

Output results (red is Design2/Design3 while Blue is Design5 with Design2 and 3 as its subclasses)**:**

```
danielrousskikh@Daniels-MacBook-Air design2 % rm PointCP2.class PointCP3.class PointCPTest.class
danielrousskikh@Daniels-MacBook-Air design2 % javac PointCP2.java PointCP3.java PointCPTest.java
danielrousskikh@Daniels-MacBook-Air design2 % java PointCPTest
Cartesian-Polar Coordinates Conversion Test Program
Testing for PointCP2 (Design 2)...
Time for creating random instances: 559 ms
Time for creating Cartesian instances: 647 ms
Time for creating Polar instances: 336 ms
Testing with Cartesian Coordinates:
Time for getX method: 494 ms
Time for getY method: 460 ms
Time for getRho method: 20 ms
Time for getTheta method: 20 ms
Time for getDistance method: 1929 ms
Time for rotatePoint method: 1303 ms
Testing with Polar Coordinates:
Time for getX method: 509 ms
Time for getY method: 466 ms
Time for getRho method: 29 ms
Time for getTheta method: 31 ms
Time for getDistance method: 1929 ms
Time for rotatePoint method: 1303 ms
Time for test to complete: 10056 ms
Testing for PointCP3 (Design 3)...
Time for creating random instances: 919 ms
Time for creating Cartesian instances: 346 ms
Time for creating Polar instances: 1244 ms
Testing with Cartesian Coordinates:
Time for getX method: 44 ms
Time for getY method: 45 ms
Time for getRho method: 47 ms
Time for getTheta method: 415 ms
Time for getDistance method: 52 ms
Time for rotatePoint method: 1929 ms
Testing with Polar Coordinates:
Time for getX method: 57 ms
Time for getY method: 59 ms
Time for getRho method: 59 ms
Time for getTheta method: 359 ms
Time for getDistance method: 65 ms
Time for rotatePoint method: 1919 ms
Time for test to complete: 7565 ms
danielrousskikh@Daniels-MacBook-Air design2 % cd ../design5
danielrousskikh@Daniels-MacBook-Air design5 % rm *.class
danielrousskikh@Daniels-MacBook-Air design5 % javac PointCP*
danielrousskikh@Daniels-MacBook-Air design5 % java PointCPTest
Cartesian-Polar Coordinates Conversion Test Program
Testing for PointCP2 (Design 2)...
Time for creating random instances: 560 ms
Time for creating Cartesian instances: 643 ms
Time for creating Polar instances: 328 ms
Testing with Cartesian Coordinates:
Time for getX method: 498 ms
Time for getY method: 461 ms
Time for getRho method: 20 ms
Time for getTheta method: 21 ms
Time for getDistance method: 1919 ms
Time for rotatePoint method: 1303 ms
Testing with Polar Coordinates:
Time for getX method: 495 ms
Time for getY method: 461 ms
Time for getRho method: 28 ms
Time for getTheta method: 30 ms
Time for getDistance method: 1921 ms
Time for rotatePoint method: 1303 ms
Time for test to complete: 10010 ms
Testing for PointCP3 (Design 3)...
Time for creating random instances: 918 ms
Time for creating Cartesian instances: 349 ms
```

```
Time for creating Polar instances: 1243 ms
Testing with Cartesian Coordinates:
Time for getX method: 44 ms
Time for getY method: 45 ms
Time for getRho method: 47 ms
Time for getTheta method: 449 ms
Time for getDistance method: 52 ms
Time for rotatePoint method: 441 ms
Testing with Polar Coordinates:
Time for getX method: 57 ms
Time for getY method: 58 ms
Time for getRho method: 60 ms
Time for getTheta method: 366 ms
Time for getDistance method: 66 ms
Time for rotatePoint method: 370 ms
Time for test to complete: 4569 ms
danielrousskikh@Daniels-MacBook-Air design5 % cd ../design2
danielrousskikh@Daniels-MacBook-Air design2 % java PointCPTest
Cartesian-Polar Coordinates Conversion Test Program
Testing for PointCP2 (Design 2)...
Time for creating random instances: 563 ms
Time for creating Cartesian instances: 634 ms
Time for creating Polar instances: 329 ms
Testing with Cartesian Coordinates:
Time for getX method: 569 ms
Time for getY method: 589 ms
Time for getRho method: 20 ms
Time for getTheta method: 21 ms
Time for getDistance method: 2130 ms
Time for rotatePoint method: 1583 ms
Testing with Polar Coordinates:
Time for getX method: 495 ms
Time for getY method: 461 ms
Time for getRho method: 29 ms
Time for getTheta method: 31 ms
Time for getDistance method: 2136 ms
Time for rotatePoint method: 1298 ms
Time for test to complete: 10911 ms
Testing for PointCP3 (Design 3)...
Time for creating random instances: 916 ms
Time for creating Cartesian instances: 335 ms
Time for creating Polar instances: 1241 ms
Testing with Cartesian Coordinates:
Time for getX method: 43 ms
Time for getY method: 45 ms
Time for getRho method: 47 ms
Time for getTheta method: 434 ms
Time for getDistance method: 53 ms
Time for rotatePoint method: 1927 ms
Testing with Polar Coordinates:
Time for getX method: 58 ms
Time for getY method: 58 ms
Time for getRho method: 60 ms
Time for getTheta method: 359 ms
Time for getDistance method: 65 ms
Time for rotatePoint method: 1936 ms
Time for test to complete: 7581 ms
danielrousskikh@Daniels-MacBook-Air design2 % cd ../design5
danielrousskikh@Daniels-MacBook-Air design5 % java PointCPTest
Cartesian-Polar Coordinates Conversion Test Program
Testing for PointCP2 (Design 2)...
Time for creating random instances: 559 ms
Time for creating Cartesian instances: 639 ms
Time for creating Polar instances: 342 ms
Testing with Cartesian Coordinates:
Time for getX method: 507 ms
Time for getY method: 457 ms
Time for getRho method: 20 ms
Time for getTheta method: 21 ms
Time for getDistance method: 1909 ms
Time for rotatePoint method: 1299 ms
Testing with Polar Coordinates:
Time for getX method: 511 ms
Time for getY method: 461 ms
Time for getRho method: 29 ms
Time for getTheta method: 30 ms
Time for getDistance method: 1931 ms
Time for rotatePoint method: 1304 ms
Time for test to complete: 10038 ms
Testing for PointCP3 (Design 3)...
Time for creating random instances: 927 ms
```

```
Time for creating Cartesian instances: 338 ms
Time for creating Polar instances: 1245 ms
Testing with Cartesian Coordinates:
Time for getX method: 44 ms
Time for getY method: 44 ms
Time for getRho method: 47 ms
Time for getTheta method: 427 ms
Time for getDistance method: 52 ms
Time for rotatePoint method: 418 ms
Testing with Polar Coordinates:
Time for getX method: 57 ms
Time for getY method: 58 ms
Time for getRho method: 60 ms
Time for getTheta method: 366 ms
Time for getDistance method: 66 ms
Time for rotatePoint method: 364 ms
Time for test to complete: 4518 ms
danielrousskikh@Daniels-MacBook-Air design5 % cd ../design2
danielrousskikh@Daniels-MacBook-Air design2 % java PointCPTest
Cartesian-Polar Coordinates Conversion Test Program
Testing for PointCP2 (Design 2)...
Time for creating random instances: 561 ms
Time for creating Cartesian instances: 663 ms
Time for creating Polar instances: 338 ms
Testing with Cartesian Coordinates:
Time for getX method: 567 ms
Time for getY method: 586 ms
Time for getRho method: 20 ms
Time for getTheta method: 21 ms
Time for getDistance method: 2138 ms
Time for rotatePoint method: 1565 ms
Testing with Polar Coordinates:
Time for getX method: 508 ms
Time for getY method: 463 ms
Time for getRho method: 29 ms
Time for getTheta method: 31 ms
Time for getDistance method: 2140 ms
Time for rotatePoint method: 1303 ms
Time for test to complete: 10952 ms
Testing for PointCP3 (Design 3)...
Time for creating random instances: 918 ms
Time for creating Cartesian instances: 345 ms
Time for creating Polar instances: 1243 ms
Testing with Cartesian Coordinates:
Time for getX method: 43 ms
Time for getY method: 45 ms
Time for getRho method: 47 ms
Time for getTheta method: 424 ms
Time for getDistance method: 53 ms
Time for rotatePoint method: 1931 ms
Testing with Polar Coordinates:
Time for getX method: 58 ms
Time for getY method: 58 ms
Time for getRho method: 60 ms
Time for getTheta method: 363 ms
Time for getDistance method: 65 ms
Time for rotatePoint method: 1925 ms
Time for test to complete: 7582 ms
danielrousskikh@Daniels-MacBook-Air design2 % cd ../design5
danielrousskikh@Daniels-MacBook-Air design5 % java PointCPTest
Cartesian-Polar Coordinates Conversion Test Program
Testing for PointCP2 (Design 2)...
Time for creating random instances: 561 ms
Time for creating Cartesian instances: 639 ms
Time for creating Polar instances: 336 ms
Testing with Cartesian Coordinates:
Time for getX method: 501 ms
Time for getY method: 464 ms
Time for getRho method: 20 ms
Time for getTheta method: 21 ms
Time for getDistance method: 1940 ms
Time for rotatePoint method: 1306 ms
Testing with Polar Coordinates:
Time for getX method: 494 ms
Time for getY method: 464 ms
Time for getRho method: 29 ms
Time for getTheta method: 30 ms
Time for getDistance method: 1933 ms
Time for rotatePoint method: 1315 ms
Time for test to complete: 10073 ms
Testing for PointCP3 (Design 3)...
```

```
Time for creating random instances: 921 ms
Time for creating Cartesian instances: 341 ms
Time for creating Polar instances: 1247 ms
Testing with Cartesian Coordinates:
Time for getX method: 43 ms
Time for getY method: 45 ms
Time for getRho method: 47 ms
Time for getTheta method: 369 ms
Time for getDistance method: 52 ms
Time for rotatePoint method: 361 ms
Testing with Polar Coordinates:
Time for getX method: 56 ms
Time for getY method: 57 ms
Time for getRho method: 60 ms
Time for getTheta method: 376 ms
Time for getDistance method: 65 ms
Time for rotatePoint method: 367 ms
Time for test to complete: 4411 ms
danielrousskikh@Daniels-MacBook-Air design5 %
```

Discussion of my results:

Initially, I assessed each design to have its pros and cons. For example, Design 1 seemed adaptable but had the overhead of a flag and conditional logic. Design 2 prioritized polar coordinates, making it faster for those but less efficient and accurate for cartesian coordinates. Design 3 was the opposite of Design 2, favoring cartesian coordinates. Design 4 aimed to eliminate any conversion loss by storing both coordinate types, at the cost of more memory. Design 5, with its inheritance structure, promised specialized behavior but at the potential cost of added complexity and slower dynamic type checking.

The tests showed that Design 5 ended up offering a large performance advantage, particularly for PointCP3 (Design 3 as subclass), it finished the tests in considerably less time compared to PointCP2 (Design 2 as subclass). Design 5 allows for more specialized behavior depending on the subclass, living up to its initial promise of faster storage and no conversion loss. PointCP2, running under Design 5, also showed slight improvements in performance, although it still takes more time to complete the tests compared to PointCP3. Overall, the results indicate that while each design has its pros and cons, Design 5 seems to offer the best trade-off between complexity and performance.

Part 2

Construction Time (in ms)

Run #	**ArrayList**	**Vector**	**Array**
Run 1	943	1433	5296

Run 2	916	1409	4640
Run 3	922	1452	4515
Run 4	932	1433	5077
Run 5	918	1427	4128
Average	926.2	1430.8	4731.2

Iteration Time (in ms)

Run #	**ArrayList**	**Vector**	**Array**
Run 1	85	219	36
Run 2	94	293	39
Run 3	99	221	36
Run 4	140	223	37
Run 5	189	219	41
Average	121.4	235.0	37.8

Conclusions

Construction Time: On average, constructing an array takes the longest time (4731.2 ms), followed by a Vector (1430.8 ms) and then an ArrayList (926.2 ms). This suggests that ArrayList is the most efficient in terms of construction time.

Iteration Time: For iterating over the collections, the array is the fastest (37.8 ms on average), followed by ArrayList (121.4 ms) and then Vector (235 ms). This indicates that if iteration is a primary concern, using an array would be the most efficient.

Recommendations

For Fast Construction: Use ArrayList, as it offers the quickest construction time among the three.

For Fast Iteration: Use a simple array, as it has the least time complexity for iteration tasks.

Dynamic Sizing: If you need dynamic sizing with good performance, ArrayList is a balanced choice.