E26 Create a table describing the various advantages (pros) and disadvantages (cons) of each of the five design alternatives. Some of the factors to consider are: simplicity of code, efficiency when creating instances, efficiency when

Table 2.1 Alternative designs for the `PointCP` class

	How Cartesian coordinates are computed	*How polar coordinates are computed*
Design 1: Store one type of coordinates using a single pair of instance variables, with a flag indicating which type is stored	Simply returned if Cartesian is the storage format, otherwise computed	Simply returned if polar is the storage format, otherwise computed
Design 2: Store polar coordinates only	Computed on demand, but not stored	Simply returned
Design 3: Store Cartesian coordinates only	Simply returned	Computed on demand, but not stored
Design 4: Store both types of coordinates, using four instance variables	Simply returned	Simply returned
Design 5: Abstract superclass with designs 2 and 3 as subclasses	Depends on the concrete class used	Depends on the concrete class used

doing computations that require both coordinate systems, and amount of memory used.

Design	Advantages (Pros)	Disadvantages (Cons)
Design 1	Flexibility to store either coordinate type.	Increased memory usage due to an extra flag, complexity in code
Design 2	Simple and memory-efficient for polar coordinates.	Inefficient for computations requiring Cartesian coordinates.
Design 3	Simple and memory-efficient for Cartesian coordinates.	Inefficient for computations requiring polar coordinates.
Design 4	Provides direct access to both coordinate types.	Uses more memory, slightly more complex code.
Design 5	Common methods shared among subclasses, code reusability.	Requires subclass-specific implementations, slight overhead.

E28 Run a performance analysis in which you compare the performance of Design 5, as you implemented it in the previous exercise, with Design 1. Determine the magnitude of the differences in efficiency, and verify the hypotheses you developed in E26.

By editing the PointCPTest.java file, we ran the program testing, taking the time to create points (same value) by using two designs (1 and 5).
Turns out Design 1 takes 6293800 nanoseconds, in comparison to design 5, which only costs 1266700. It indeed fits our hypotheses.

Run Results Shown

```
******************************Design 1*****************************************
Enter the type of Coordinates you are inputting ((C)artesian / (P)olar): P
Enter the value of Rho using a decimal point(.): 2
Enter the value of Theta using a decimal point(.): 3

You entered:
Stored as Polar [2.0,3.0]


After asking to store as Cartesian:
Stored as Cartesian  (1.9972590695091477,0.10467191248588767)


After asking to store as Polar:
Stored as Polar [1.9999999999999998,3.0000000000000004]


The performance of the Design1 shows a construction time of 6293800 nanoseconds.

******************************Design 5*****************************************
Enter the Rho using a decimal point(.):
2
Enter the Theta using a decimal point(.):
3

You entered:
Stored polar coordinates are (2.0,3.0)

After asking to create a now point stored as Cartesian:
Stored Cartesian coordinates are (1.9972590695091477, 0.10467191248588767)

After asking to create a now point (again) stored as Polar:
Stored polar coordinates are (1.9999999999999998,3.0000000000000004)

The performance of the Design5 shows a construction time of 1266700 nanoseconds.
ANALYZING....
...
Turns out Design 5 is a better design in an efficient manner.

Process finished with exit code 0
```

E29 To run a performance analysis, you will have to create a new test class that randomly generates large numbers of instances of `PointCP`, and performs operations on them, such as retrieving polar and Cartesian coordinates. You should then run this test class with the two versions of `PointCP` – Design 1 and Design 5.

E30 Summarize your results in a table: the columns of the table would be the two designs; the rows of the table would be the operations. The values reported in the table would be the average computation speed. Make sure you explain your results.

Operations (10^8 times)	Design 1 (in seconds)	Design 5 PointCP2 (in seconds)	Design 5 PointCP3 (in seconds)
Create new Point Variables	6.546162999	6.0682668	6.007594001
convertStorage() for Design 5, just swap the CP2 and Cp3	7.9972652	6.2610311	5.9523062
rotatePoint()	0.019483199	0.0110027	0.011949701
getDistance() by creating random points	3.404087099	3.3480851	3.354192801

```
C:\Users\Roy\.jdks\graalvm-jdk-17.0.8\bin\java.exe "-javaagent:A:\IntelliJ IDEA\lib\idea_rt.jar=51758:A:\IntelliJ IDEA

***DESIGN 1: The construction time for creating given number 100000000 of points are 6.546162999 seconds.

***DESIGN 5 CP2: The construction time for creating given number 100000000 of points are 6.0682668 seconds.

***DESIGN 5 CP3:The construction time for creating given number 100000000 of points are 6.007594001 seconds.

Process finished with exit code 0
```

```
C:\Users\Roy\.jdks\graalvm-jdk-17.0.8\bin\java.exe "-javaagent:A:\IntelliJ IDEA\lib\idea_rt.jar=51879:A:

***DESIGN 1: The construction time for converting 100000000 times are 7.9972652 seconds.

***DESIGN 5 CP2: The construction time for converting 100000000 times are 6.2610311 seconds.

***DESIGN 5 CP3:The construction time for converting 100000000 times are 5.9523062 seconds.

Process finished with exit code 0
```

```
***DESIGN 1: The construction time for rotating 100000000 times are 0.019483199 seconds.

***DESIGN 5 CP2: The construction time for rotating 100000000 times are 0.0110027 seconds.

***DESIGN 5 CP3:The construction time for rotating 100000000 times are 0.011949701 seconds.

Process finished with exit code 0
```

```
C:\Users\Roy\.jdks\graalvm-jdk-17.0.8\bin\java.exe "-javaagent:A:\IntelliJ IDEA\lib\idea_rt.jar=51959:A:\Intell

***DESIGN 1: The construction time for measuring distances 100000000 times are 3.404087099 seconds.

***DESIGN 5 CP2: The construction time for measuring distances 100000000 times are 3.3480851 seconds.

***DESIGN 5 CP3:The construction time for measuring distances 100000000 times are 3.354192801 seconds.

Process finished with exit code 0
```

By creating a new class named PerformanceTest.java in the directory of part 1, and experimenting with the performance of two different designs, we can see that in the table, Design 1 shows weakness at every extent.

The difference between operating actions like creating new points, and methods like rotatePoint() and getDistance() seems to be small enough to get ignored.

However, on the speaking of converting storage of object data (x and y, or rho and theta), Design 1 is way bad for Design 5: by only 100000000 operating, Design 1 gives almost 2 seconds lag in comparison of Design 5. It is time-consuming indeed.

Kindly notice that in part 2, there might be hardware issues related to the memory heap when running, it would be hard for some old computers to get computed successfully. Please contact us if the mentioned issue happens.