E25:

a) The methods: convertStorageToPolar and convertStorageToCartesian are used to convert the equations to polar or Cartesian. This equation can be expressed more intuitively in Cartesian form. The polar form directly presents the degree between the x-axis and the point. If the appropriate form of equation is used under different requirements, the performance of the program can be significantly improved. So, the program needs some methods to convert equations to another form.

b) The program may frequently call two methods to convert the equations to another form. In this situation, the program may take longer time than expected.

c) Code:

```
  long start = System.currentTimeMillis();
  for (int s = 0; s < 10001; s++){
    point.getX();
    point.getY();
  }
  long end = System.currentTimeMillis();
  System.out.println("Elapsed time: " + (end - start));

  start = System.currentTimeMillis();
  for (int i = 0; i < 10001; i++) {

    //System.out.println("\nYou entered:\n" + point);
    point.convertStorageToCartesian();
    //System.out.println("\nAfter asking to store as Cartesian:\n" + point);
    point.convertStorageToPolar();
    //System.out.println("\nAfter asking to store as Polar:\n" + point);

  }
  end = System.currentTimeMillis();
  System.out.println("Elapsed time: " + (end - start));
}
```

Result:

```
C:\Users\xiang\.jdks\openjdk-22.0.1\bin\java.exe "-javaagent:D:\jetbrains\Programs\IntelliJ IDEA Ultimate\lib\idea_rt.jar=63521:D:\jetbrains\Programs\IntelliJ IDEA U
Cartesian-Polar Coordinates Conversion Program
Enter the type of Coordinates you are inputting ((C)artesian / (P)olar): c
Enter the value of X using a decimal point(.): 1.0
Enter the value of Y using a decimal point(.): 1.0
Elapsed time: 0
Elapsed time: 2

进程已结束，退出代码为 0
```

I wrote a small program to explore its weaknesses. The trivial conversion caused it to take 2 milliseconds longer.

E26:

Design	How cartesian coordinates are computed	How polar coordinates are computed
Design2: Store polar coordinates only	The calculation is simple, and the procedure takes very little time.	The program does frequent format conversions which can use a lot of time.
Design3: Store polar coordinates only	The program does frequent format conversions which can use a lot of time.	The calculation is simple, and the procedure takes very little time.
Design5: Store polar coordinates only	depend on the program used.	depend on the program used.

E30:

Design 2 and 3:

Design	cartesian coordinates 2000000000 times	polar coordinates 2000000000 times
Design2	5 mills	16364 mills
Design3	12582 mills	3 mills

Design 5:

Design(subclass use)	cartesian coordinates 2000000000 times	polar coordinates 2000000000 times
Design2	6 mills	15462 mills
Design3	13584 mills	2 mills

Design Method 2 excels in Cartesian coordinates with a performance of 5 milliseconds for 2 billion calculations, but it is less efficient with polar coordinates, taking 16,364 milliseconds for the same number of calculations. Conversely, Design Method 3 is optimized for polar coordinates, requiring only 3 milliseconds for 2 billion calculations, yet it takes 12,582 milliseconds for Cartesian coordinates. Design Method 5 combines these two approaches using an abstract class, offering flexibility by leveraging the strengths of both methods. This combined method takes 6 milliseconds (Design 2) and 13,584

milliseconds (Design 3) for Cartesian coordinates, and 15,462 milliseconds (Design 2) and 2 milliseconds (Design 3) for polar coordinates. Furthermore, Design Method 5 is highly adaptable for future modifications, making it particularly suitable for evolving requirements and enhancements. While it provides versatility and optimizes performance across different scenarios, it also introduces additional complexity and potential overhead due to its combined nature.