|  |  |  |
| --- | --- | --- |
| Design | Pros | Cons |
| Design 1 | Memory usage is better and so more efficient for class allocation | Not efficient and may lose precision after multiple conversions. |
| Design2 | More efficient for and cartesian polar conversion | The con is that it deals with both polar and cartesian (it should only take care of polar)  Less efficient for allocation for cartesian input |
| Design3 | More efficient for cartesian and polar conversion | The con is that it deals with both polar and cartesian (it should only take care of Cartesian)  Less efficient for allocation for polar input |
| Design4 | Most efficient for conversion | Memory hungry, it uses twice the memory.  And so least efficient for allocation overall |
| Design5 | Most efficient overall and has clear separation between both polar and cartesian representations. Also has more rigor in terms of code and is more up to standard of what a software engineer should produce as a product. | Need to allocate the right subclass. |

**E26**

**E30**

|  |  |  |  |
| --- | --- | --- | --- |
| Design | Conversion | Polar allocation | Cartesian allocation |
| Design1 | 7178 | 2 | 2 |
| Design2 | 5 | 4\* | 5079 |
| Design3 | 5 | 5079\* | 4 |
| Design4 | n/a | n/a | n/a |
| Design5 | 5 | 9\* | 9 |

\* Values were determined from cartesian results.

All these values are in milliseconds.

**E25**

1. Some operations are quicker and radiance and other operations are quicker in cartesian.
2. It will keep on converting therefore there will be a massive efficiency problem
3. Done