1. Denksportaufgabe

1.1. Es wird `false` ausgegeben, weil für die Klasse Name keine `hashCode()` Methode geschrieben wurde und daher für die beiden erzeugten Instanzen von Name zwei verschiedene Hashcodes erzeugt werden. Aus diesem Grund wertet `s.contains(new Name("Mickey", "Mouse"))` zu false aus. `equals(Object o)` und `hashCode()` sind hier also nicht zueinander konsistent. `equals()` könnte aus Performancegründen noch um einen null-Check und Objektidentität-Check erweitert werden.

1.2. Hier wird nun `true` ausgegeben, da `hashCode()` korrekt implementiert ist. Allerdings wurde hier equals unzureichend implementiert. Es lässt sich nun nur die aktuelle Instanz mit einer anderen Instanz der Klasse Name vergleichen. Vergleiche mit Objekte anderen Typs werden nun mittels der equals Methode aus `Object` vorgenommen. Gibt es nun z.B. ein Objekt aus einer Klasse, die von `Name` erbt, lässt es sich nicht mit einem `Name`-Objekt vergleichen.

2. Komplexe Zahlen

Wir modellieren eine komplexe Zahl in der Klasse `Complex`.
`CoordinateCartesian` bildet schlicht eine kartesische Koordinate ab, die auch außerhalb des Kontexts der komplexen Zahlen verwendet werden kann. Gleiches gilt für eine Polarkoordinate aus `CoordinatePolar`.
Alle Klassen `Complex`, `CoordinatePolar` und `CoordinateCartesian` sind mutable, d.h sie verfügen über Methoden, die deren Werte ändern, wie `setReal`, `setImaginary`, ... usw.

Rechenoperationen, die die mutable komplexe Zahl verändern sollen, sind direkt in Complex implementiert, wie `add`, `subtract`, `multiply`, `divide`. Konversionen von/zu Koordinaten und komplexen Zahlen werden in `ComplexMath` realisiert.
Rechenoperationen mit einer komplexen Zahl, die sie selbst aber nicht verändern, sondern eine Neue zurückgeben, sind in der Utilityklasse `ComplexMath` implementiert. Die Methoden `sin`, `cos`, `tan` und `exp`, der Klasse `ComplexMath` erhalten ein Objekt der Klasse `Complex` als Parameter und geben das Ergebnis der Operation als neue Instanz von `Complex` zurück. Da eine UtilityClasse ist, hat sie die Signature `public final class ComplexMath` und hat einen privaten Konstruktor.