Interface

De klasser/filer som hänvisas till hittar du på kurssidan under ”Övningar/Lösningsförslag”.

Uppgift 1, java.lang.Comparable

För att det ska vara möjligt att jämföra kort, t ex sortera en hand, är det lämpligt att låta klassen Card implementera interfacet `Comparable<T>`, som har till uppgift att definiera den naturliga ordningen för ett objekt av en viss klass.

Sök upp detta interface i API-dokumentationen och läs om metoden `compareTo(T other)`.
Låt din kortklass implementera interfacet `Comparable`. Låt ordningen avgöras efter färg, och om färgerna är lika, efter valör.

Skriv ett main där du skapar ett antal kort i en array, osorterad, och sedan använder dig av klassmetoden `sort(Object[] o)` i klassen `Arrays` för att sortera dessa (läs i API)..
Många klasser som lagrar objekt sorterat förutsätter att objekten som lagras implementerar `Comparable`.

Om en klass implementerar `Comparable` rekommenderas att metoden `equals`, ärvd från Object, omdefinieras så att `(c1.compareTo(c2) == 0) ⇔ (c1.equals(c2) == true)`. Lägg till denna metod i kortklassen.

Om du har en klass som representerar en korthand kan du lägga till en metod för att sortera korten i denna. Se API-dokumentationen för klassen java.util.Arrays och metoden sort.

Uppgift 2, skriva ett eget interface

Skriv interfacet `GraphicsObject` som deklarerar de tre metoderna:

```
public double area()
public void translate ( double dx, double dy )
public boolean inside( double x, double y )
```

Metoden `translate` flyttar figuren/figurerna dx resp dy i x resp y-led. Metoden `inside` returnerar true om x och y ligger inom figuren.

De båda klassern, `Circle` och `Group` implementerar båda interfacet ovan. Fler klasser kan senare tillkomma, men du ska bara skriva dessa två..
Objekt av klassen `Circle` har en mittpunkt (x, y) samt en radie.
Objekt av klassen `Group` innehåller en mängd grafiska objekt, så det ska finnas en metod för att lägga till grafiska objekt. Metoden `area` returnerar i detta fall summan av alla areor i gruppen.
Metoden `inside` returnerar i detta fall true om punkten ligger inom någon av figurerna.

a) Skriv klasserna `Circle` och `Group`.
b) Skriv en klass, `GraphicsWindow`, som ska innehålla ett antal grafiska objekt objekt (`Circle` och `Group`). Klassen ska minst ha metoderna:
`add` – tar ett objekt av en klass som implementerar `GraphicsObject` och lägger in det i ”fönstret”

`remove(double x, double y)` – tar bort alla figurer som innehåller (x,y) enligt `inside`.
`area` – ger summan av alla ytor.
`translate(double dx, double dy)` – förflyttar alla figurer.
Skriv ett main som testar klasserna (information om de grafiska objekten ges med toString).

Uppgift 3, endast för den djärve

Vi har möjlighet att definiera alternativa ordningar med hjälp av interfacet
`java.util.Comparator` som innehåller metoderna
`int compare(Object o1, Object o2)` och
`boolean equals(Object obj).`

I detta fall skapar du en separat ”jämförarklass” som i vårt fall jämför 2 kort enligt någon speciell ordning.
I klasserna `Arrays` och `Collections` hittar du en version av sorteringsmetoden,
`static void sort(Object[] a, Comparator c)`, där du får sortering enligt din ”jämförare”.

Läs mer om interfacet i API-dokumentationen.

Skapa en ”jämförarklass”, `CardByValueComparator` som implementerar Comparator och sorterar kort efter valör, och om valörerna är lika efter färg[1]. Sortera samma kort som i uppgift 1 med denna ”jämförare”.

[1] Notera: Din ”jämförarklass” är bara meningsfull att använda tillsammans med klassen Card, och bör tillhöra samma paket som kortklassen. Det är lämpligt att definiera den som en (icke publik) klass i filen Card.java, alternativt som en inre klass i kortklassen. Du måste då skriva en metod som returnerar en referens till en ”jämförare” i den publika kortklassen.