Uppgift 1

I Javas standardbibliotek finns en klass för rektanglar. Nedanstående kod visar relevanta delar av klassen:

```
public class Rectangle {
    . . .
    public Rectangle(int height, int weight) {...}
    public Rectangle(int x, int y, int height, int weight) {...}
    public double getWidth() {...}
    public double getHeight() {...}
}
```

Tyvärr har denna klass inga metoder för att beräkna area och omkrets. Gör en ny klass ImprovedRectangle som ärver från klassen Rectangle och lägger till två metoder för att beräkna area och omkrets. Lägg inte till några nya instansvariabler! Klassen Rectangle finns i paketet java.awt.

Uppgift 2

Betrakta nedanstående tre klasser för att avbilda olika räknare – klassen Counter, klassen BoundedCounter och klassen ChainedCounter.

Dessa klasser är inte relaterade till varandra på något sätt, men har ju många gemensamma drag. De modellerar likartade objekt (olika slag av räknare) och de har samma uppsättning operationer (count(), getValue() och reset()).

Borde klasserna varit implementerade på något annat sätt? Motivera ditt svar! Gör implementationen.

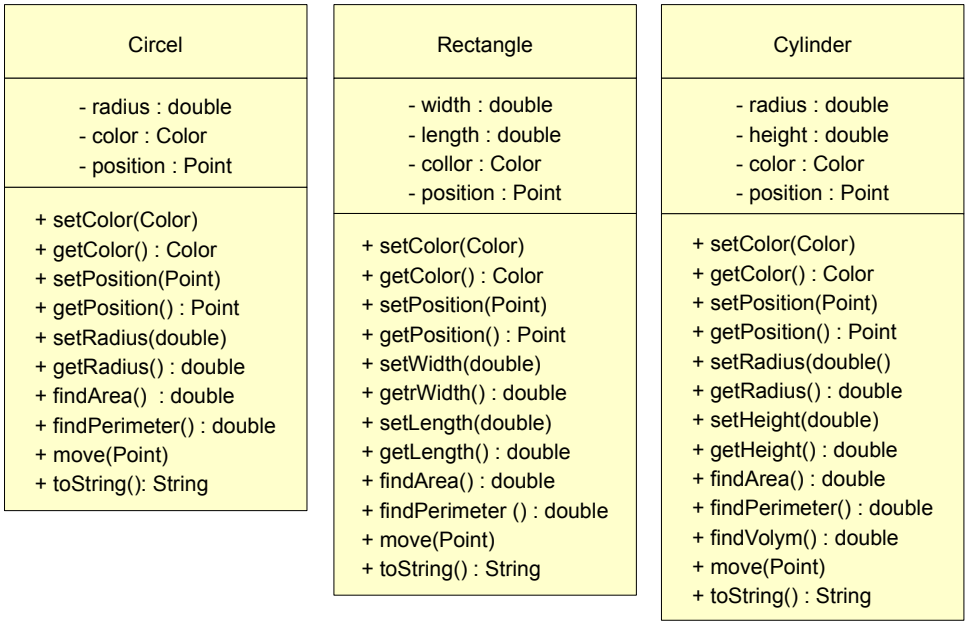
```
public class Counter {
    private int value;
    public Counter() {
        value = 0;
    }//constructor

    public void count() {
        value = value + 1;
    }//count

    public int getValue() {
        return value;
    }//getValue

    public void reset() {
        value = 0;
    }//reset
}//Counter
```

```
public class BoundedCounter {
    private int value;
    private int modulus;

    public BoundedCounter(int modulus) {
        value = 0;
        this.modulus = modulus;
    }//constructor

    public void count() {
        value = (value + 1) % modulus;
    }//count

    public int getValue() {
        return value;
    }//getValue

    public void reset() {
        value = 0;
    }//reset
}//BoundedCounter
```

```
public class ChainedCounter {
    private int value;
    private int modulus;
    private Counter next;

    public ChainedCounter(int init, int modulus, Counter next) {
        value = init;
        this.modulus = modulus;
        this.next = next;
    }//constructor

    public void count() {
        value = (value + 1) % modulus;
        if (value == 0 && next != null)
            next.count();
    }//count

    public void reset() {
        value=0;
    }//rerset

    public int getValue() {
        return value;
    }//getValue
}// ChainedCounter
```

Uppgift 3

I ett program som handhar geometriska objekt har vi identifierat följande klasser:



Som synes har klasserna en hel del gemensamt. Gör en implementera av klasserna Circel, Rectangle och Cylinder i vilken arv utnyttjas. Sträva efter att återanvända så mycket kod som möjligt.

Uppgift 4

Klassen Person används för att avbilda personer. Klassen innehåller bl.a. följande metoder:

public String getName()	returnerar namnet
public int getBirthYear()	returnerar födelseår

a) Skriva en metod

public static ArrayList<String> bornThisYear(ArrayList<Person> people, **int** year)

som tar en lista people av Person-objekt samt ett årtal year, och returnerar en lista som innehåller namnen på de personer i listan people som är födda år year.

b) Skriva en metod

public static void removeNames(ArrayList<Person> people, ArrayList<String> names)

som tar en lista people och en lista names, och tar bort alla objekt i listan people vars namn finns med i listan names.

Uppgift 5

Vi kan representera de olika färgerna i en kortlek med följande uppräkningstyp:

```
public enum Suit {
    HEARTS, SPADES, DIAMONDS, CLUBS;
}
```

De olika färgerna i en kortlek trycks ofta med färgerna svart och rött. Färgerna hjärter och ruter trycks med röd färg och spader och klöver med svart.

• Skriv en uppräkningstyp som representerar färgerna rött och svart. Kalla typen för SuitColor t.ex.

• Lägg till ett tillstånd till typen Suit så att varje färg även har en tryckfärg.

• Lägg till en metod i typen Suit som returnerar vilken tryckfärg som färgen har. Funktionen ska ha följande signatur:

public SuitColor getColor();

Uppgift 6

Du ska definiera klassen Superhero. En superhjälte är som en vanlig människa; den kan äta, dricka, och sova. Men utöver det har den även vissa superkrafter; alla superhjältar kan flyga, läka sig och rädda civila. För att få vara med i superhjälteligan måste din superhjälte implementera det interface som finns för superhjältar. Du behöver göra följande:

Skapa klassen Human, som har det beteende beskrivet ovan.

Skapa interfacet Superpowers, som innehåller alla egenskaper en superhjälte måste ha.

Skapa klassen SuperHero, som är en utökning av Human, men även implementerar Superpowers.

Varje ”beteende” kommer i det här fallet helt enkelt att vara en metod, ex. sleep(**int** minutes). Du behöver dock inte skriva någon kod i själva metoderna, du kan lämna dem tomma.

Uppgift 7 (NY)

Denna uppgift handlar om båtar som skall sättas i sjön av en båtklubb.

Båtar som ska sjösättas har en viss vikt och ett visst djupgående. För att sjösättas måste båten vara förberedd, generellt gäller att båten måste vara bottenmålad (det innebär att ägaren har målat båtens botten med en skyddsfärg) men sedan tillkommer det lite olika krav för olika båttyper (se nedan). Dessutom får inte båten vara för tung för båtklubbens kran.

För sjösättningen tar båtklubben en avgift, avgiftens storlek beräknas på olika sätt för olika båttyper men man bör se till att avgiften betalas endast om båten blivit sjösatt - om båtklubben begär avgiften för en båt som inte har blivit sjösatt får den 0 kronor.

Båtar kan vara Motorbåtar eller Segelbåtar. En Motorbåts vikt är olika från båt till båt, men djupgående är alltid 70 (cm). Förutom att vara bottenmålad måste en Motorbåt ha bränsle för att vara klar för sjösättning, så man ska kunna tanka en Motorbåt med bränsle. Sjösättningsavgiften för en Motorbåt är 1000 kr om båten väger mindre än 2000 kg, annars 1500 kr.

För en Segelbåt varierar både vikten och djupgåendet från båt till båt. En Segelbåt måste (förutom att vara bottenmålad) vara påmastad (ha masten på) för att få sjösättas. Segelbåten ska därför kunna mastas på. Sjösättningsavgiften beräknas som vikt gånger djupgående.

Båtklubben representeras av följande klasskelett:

```
public class Båtklubb {
  private int pengar = 0, maxVikt;

  public Båtklubb(int maxVikt){
    this.maxVikt = maxVikt;
  }
  public void sjösättning(ArrayList<Båt> påLand, ArrayList<Båt> iSjön){
    // Metodkroppen ska du skriva
  }
  public int getPengar(){
    return pengar;
  }
}
```

maxVikt är den maximala vikt båtklubbens kran kan lyfta, pengar är summan av avgifter båtklubben tagit för sjösättning av båtarna. Metoden sjösättning ska som argument få dels en ArrayList påLand som innehåller blandade båt-objekt, och dels en ArrayList iSjön som från början är tom. Metoden ska titta på varje båt i samlingen påLand och om båten inte är för tung och är klar för sjösättningen så ska den flyttas från samlingen påLand till samlingen iSjön och sjösättningsavgiften ska läggas till båtklubbens pengar.

Skriv klasserna Båt, Motorbåt och Segelbåt (med attribut, konstruktorer och metoder) samt Båtklubbens metod sjösättning. Attributen i klasserna ska vara privata.