Exercise 1: Write a class `Ship`, which as fields `name` and `length`. Write a subclass `ContainerShip` of class `Ship` which has a field `teu` (Twenty-foot Equivalent Unit). Write a subclass `Tanker` of class `Ship` which has a field `barrels`. Write a subclass `CruiseLiner` of class `Ship` which has a field `noOfPassengers`. The fields should be strings or integers as appropriate. Make all fields private and implement getter-methods. Also provide the constructors. Provide `toString`-methods.
On CodeJudge you find a test class which expects that the `toString` in class `Ship` is implemented by

```
public String toString(){
        return "Ship "+name+" l="+length;
}
```

and in `ContainerShip` by

```
public String toString(){
        return super.toString()+"  container carrier with "+teu+" TEU";
    }
```

and similar for the other subclasses.

End of Exercise 1

Exercise 2: Write class `Vessel` with a field `volume`. Write a subclass `Bottle` of `Vessel` with a field `content`. Write a subclass `GlasBottle` of `Bottle` with a field `color`. Write a subclass `PlasticBottle` of `Bottle` with a field `material` (values can be PET or OTHER).
On Campusnet you find a class `Shop` which tests your implementation. The same class is used on CodeJudge, but you have to ensure that your `toString` methods give the output expected by CodeJudge.

End of Exercise 2

Exercise 3: På CodeJudge under 8.3 finder du skellettet til klassen `Spiller`, som nedarver egenskaber fra klassen `Point`.

- **Constructor.** Lav en constructor `Spiller(int x, int y, int n)` som sætter banens størrelse til n og placerer spilleren på (x, y). Hvis banens størrelse er (svagt) negativ eller spilleren er udenfor brættet, skal I give en `Exception`.
- **toString.** Lav en offentlig metode `toString()` som returnerer strengen indeholdende punktets koordinater på følgende form: `"[x;y]"`

  altså f.eks. `"[3;4]"`.
- **translate.** Lav en offentlig metode `translate(int dx, int dy)`, der flytter spilleren med (dx, dy), dog ikke udenfor brættet.

End of Exercise 3

Exercise 4: Lav klassen `InstansAf` som har den offentlige metode `kunStrengene(Object[] objekter)`, som printer strengene i arrayet `objekter`, *i rækkefølge*, separeret af linieskift. Hvis arrayet `objekter` indeholder objekter som ikke er strenge, skal disse ignoreres, og altså hverken printes eller føre til linieskift.
Eksempel:

```
Object[] eksempel = {"hej", new Point(42,0), "med dig"};
InstansAf.kunStrengene(eksempel);
```

giver

```
hej
med dig
```

End of Exercise 4

Exercise 5: På CodeJudge ligger et skellet til klassen `Springer extends Point`, som er et punkt, der kun kan translateres som en springer i et skakspil: To felter frem og én til siden. (Ialt findes der otte felter, den kan nå).
Lav den offentlige metode `translate(int dx, int dy)`, som tester om translationen er lovlig for en springer, og i så fald udfører den, og som ellers lader "skak"brikken stå hvor den er.
Benyt `super.metodenavn` til at forsimple din kode.

End of Exercise 5