

Opgave 1.

```
a) interface BagInterface {
    /* Elementen: objecten van het type X
       Structuur: geen
       domein : alle groepen van elementen van het type X
       waarbij dubbelen mogen voorkomen
    */

    /* Bag-PRE en Bag-POST zijn de waarden van de bag ten tijde
       van respectievelijk de PRE- en POST-conditie.
    */

    /*
    Bag ();
    PRE -
    POST - Een leeg Bag-object is aangemaakt.
    */

    void voegToe (X x);
    /* PRE -
       POST - Een kopie van x is aan Bag-PRE toegevoegd.
    */

    boolean verwijder (X x);
    /* PRE -
       POST - FALSE: x zat niet in Bag-PRE
              TRUE : Het element x is 1 keer uit Bag-PRE verwijderd.
    */

    boolean size ();
    /* PRE -
       POST - Het aantal elementen in de Bag is geretourneerd.
    */

    public Object clone ();
    /* PRE -
       POST - Een deep copy van Bag is geretourneerd.
    */

}
```

```
b) class Bag implements BagInterface {

    static final int DEFAULT_SIZE = 20;

    private X[] rij;
    private int aantalElementen;
```

```
public Bag () {
    rij = new X[DEFAULT_SIZE];
    aantalElementen = 0;
}

public void voegToe (X x) {
    if (aantalElementen == rij.length) {
        verdubbelRijlengte();
    }

    rij[aantalElementen] = x.clone();
    aantalElementen += 1;
}

private verdubbelRijlengte () {
    int[] resultaat = new int[rij.length * 2];
    for (int i = 0; i < rij.aantal; i++) {
        resultaat[i] = rij[i];
    }

    rij = resultaat;
}

public boolean verwijder (X x) {
    boolean gevonden = false;
    int index;

    for (int i=0; i<aantalElementen && !gevonden; i++) {
        if (rij[i].equals(x)) {
            gevonden=true
            index=i;
        }
    }

    if (index != -1) {
        rij[index] = rij[aantalElementen-1];
        aantalElementen -= 1;
    }

    return gevonden;
}

public int size () {
    return aantalElementen;
}

public Object clone () {
    Bag kopie;
```

```

    try {
        kopie = (Bag) super.clone();
    } catch (CloneNotSupportedException e) {
        throw new Error("object heeft niet type Cloneable");
    }

    kopie.rij = new X[rij.length];
    for (int i=0; i<aantalElementen; i++) {
        kopie.rij[i] = (X) rij[i].clone();
    }

    return kopie;
}
}

```

Opgave 2.

- a) CardinalVerzameling () {
 super(MAX_AANTAL_ELEMENTEN);
 }

 public Object clone() {
 return super.clone();
 }
- b) void voegToe (int getal) {
 if (getal < 0) {
 return; // negeren
 // throw new Error("FOUT"); als alternatief: foutmelding
 }

 super.voegToe(getal);
 }
- c) public boolean equals (Object rhs) {
 if (rhs == this) {
 return true;
 }

 if (rhs == null) { // N.B. Deze test is eigenlijk niet nodig
 return false; // omdat bij ontbreken ervan het onderstaande
 } // if-statement dit geval opvangt.

 if (! (rhs instanceof IntVerzameling)) {
 return false;
 }
 }

```
IntVerzameling set = (IntVerzameling) rhs;
```

```
// Zitten alle elementen van this in set?
for (int i=0; i<aantalElementen; i++) {
    if (! set.isElement(intArray[i])) {
        return false;
    }
}

// Zitten ale elementen van set in this?
for (int i=0; i<zet.aantalElementen; i++) {
    if (! isElement(zet.intArray[i])) {
        return false;
    }
}

return true;
}
```

- d) Zo'n lege interface heet een 'marker interface'

Het enige doel van een markerinterface is om een type te definiëren.

Dit type kan worden gebruik om

- 1- een generieke datastructuren te maken waarbij het aantal classes waarvan instanties kunnen worden gebruikt beperkt wordt tot die classes die de marker interface implementeren.
- 2- aan te geven dat bepaalde bewerkingen mogen plaatsvinden. Bijvoorbeeld de methode clone() die alleen werkt voor objecten met het type Cloneable.

- e) De wrapper classes zijn classes zoals Integer en Character die niets anders bevatte dan 1 waarde van een primitief type. Het doel van deze classes is om een primitief type als het ware om te zetten in een object-type zodat het mogelijk is dit primitief type mee te geven aan een generieke datastructuur.

- f) static final int MAX_AANTAL_ELEMENTEN = 75;

```
IntVerzameling[] verzamlingRij=new
IntVerzameling[MAX_AANTAL_ELEMENTEN];
```