Opgave 10

Følgende er en abstrakt klasse for repræsentation af matricer i JAVA;

```
abstract class Matrix {
  public int rows, columns;

  // get returnerer indholdet af indgangen i række i og søjle j
  abstract public int get(int i,int j);

  // set sætter indholdet af indgangen i række i og søjle j til v
  abstract public void set(int i, int j, int v);

  // multiply returnerer matrix-produktet af denne matrix og søjlvektoren x
  abstract public Matrix multiply(Matrix x) throws RuntimeException;
}
```

Objekter af klassen Matrix kan repræsenteres ved hjælp af arrays, og en implementation kunne se ud som følger:

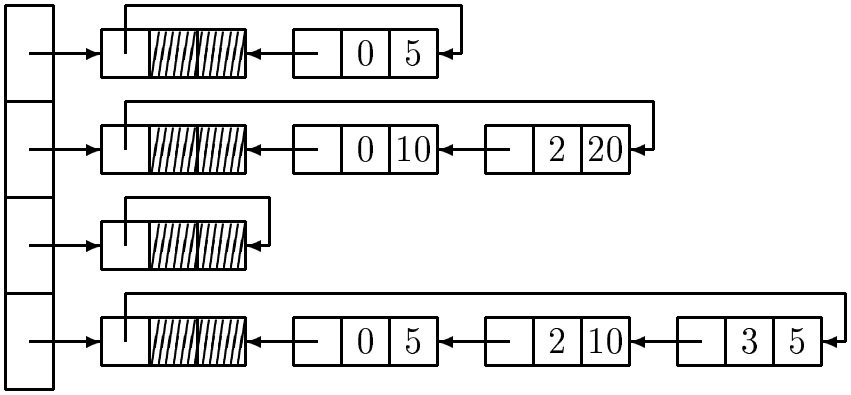
```
class ArrayMatrix extends Matrix {
  private int[][] M;

  ArrayMatrix(int r, int c) {
    rows=r;
    columns=c;
    M = new int[r][c];
  }

  public int get(int i, int j) {
    return M[i][j];
  }

  public void set(int i, int j, int v) {
    M[i][j]=v;
  }

  public Matrix multiply(Matrix x) throws RuntimeException {
    ArrayMatrix result;
    if (x.rows != columns || x.columns!=1)
      throw new RuntimeException("Incompatible dimensions for multiply.");
    else {
      result = new ArrayMatrix(1,rows);
      for (int i=0; i<rows; i++) {
        result.set(i,0,0);
        for (int j=0; j<rows; j++)
          result.set(i,0, result.get(i,0) + get(i,j)*x.get(j,0));
      }
    }
    return result;
  }
}
```

En matrix (og en vektor) siges at være *tynd*, såfremt hovedparten af dens elementer er 0. Store, tynde matricer og vektorer kan repræsenteres væsentligt mere hensigtsmæssigt ved i stedet for ArrayMatric definitionen (*) at anvende en type, som gør det muligt kun at gemme de elementer, der er forskellige fra 0. Én af metoderne består i at repræsentere hver række v.hj.a. en kædet liste som illustreret ved følgende eksempel.

Matricen

$$M = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 10 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 \\ 5 & 0 & 10 & 5 \end{bmatrix}$$

repræsenteres på følgende måde



a) Beskriv en klasse TyndMatrix, der implementerer Matrix og skitser en algoritme for metoden `multiply`.

b) Antag nu, at klassen Matrix udvides med en metode, som multiplicerer to matricer:

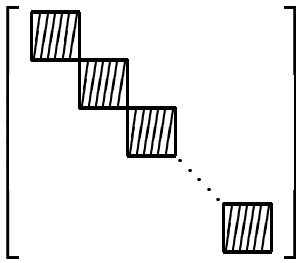
```
// product returnerer matrix-produktet af denne matrix og m
abstract public Matrix product(Matrix m) throws RuntimeException;
```

For ArrayMatrix kan en implementationen for `product` se ud som følger:

```
public Matrix product(Matrix m) throws RuntimeException {
  ArrayMatrix result;
  if (m.rows != columns)
    throw new RuntimeException("Incompatible dimensions for product.");
  else {
    result = new ArrayMatrix(rows, m.columns);
    for (int i=0; i<rows; i++) {
      for (int j=0; j<m.columns; j++) {
        result.set(i,j,0);
        for (int k=0; k<columns; k++)
          result.set(i,j, result.get(i,j) + get(i,k)*m.get(k,j));
      }
    }
  }
  return result;
}
```

Forklar hvorfor den nuværende repræsentation for TyndMatrix er uhensigtmæssig med henblik på multiplikation af tynde matricer. Angiv en mere hensisgtmæssig repræsentation og skitser, hvordan en effektiv implementation for `product` for TyndMatrix virker.

c) Antag at M1 og M2 er to (såkaldt *blokdiagonale*) $n \times n$ matricer af følgende form



hvor alle elementer uden for de skraverede blokke er 0; og hvor der er ialt $\sqrt{n}$ lige store blokke, som altså selv er $\sqrt{n} \times \sqrt{n}$ matricer.

Idet X er en vilkårlig vektor af længde n, skal størrelsesordenen af udførelsestiden for følgende metodekald angives.

i) `m1.product(m2)` i ArrayMatrix

ii) `m1.multiply(x)` i TyndMatrix

iii) `m1.multiply(x)` i ArrayMatrix

iv) `m1.product(m2)` i TyndMatrix

hvor `m1`, `m2` og `x` er repræsentationer af $M1$, $M2$ og X.

Angiv også størrelsesordenen af det antal multiplikationer, der udføres i de fire procedurekald.