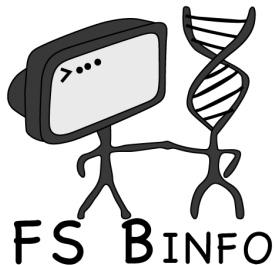


# Programmier-Aufgabe zur Wiederholung



Fabian Schmich

Fachschaft Bioinformatik

15. Dezember 2009

# Zeitplan

---

- 27.10 Einführung in Linux und die Shell
  - 03.11 Arbeiten mit Dateien auf der Shell
  - 10.11 Einführung in Java
  - 17.11 Umgang mit Arrays, Schleifen und Dateien
  - 24.11 Einführung in objekt-orientierte Programmierung
  - 01.12 Rekursion und Iteration
  - 08.12 File-I/O, Schleifen, Javadoc
  - 15.12 **Programmier-Aufgabe zur Wiederholung**
- 
- 12.01 Debugging
  - 19.01 Vererbung, Packages, allgemeine Tipps
  - 26.01 Wiederholung

Webseite: <http://www.bioinformatik-muenchen.com/bioinfocom/informatik-tutorium>



# Wiederholung

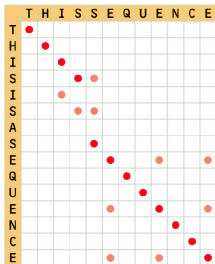
---

- Variablen deklarieren und initialisieren
- Arrays
- `if/else` Statements
- `for` und `while` Schleifen
- Klassen und Konstruktoren
- Methoden
- Einfacher File Input- und Output



# Dotplots

Schneller und einfacher visueller Sequenzvergleich (vgl. Vorlesung Einführung in die Bioinformatik, 11. Dezember)



(Zvelebil & Baum)

- Auftragen der Sequenzen an X- und Y-Achse
- Punkt an den Stellen zeichnen, an denen Sequenzen sich gleichen
- Identifizierung von Sequenzmerkmalen wie z.B. In/Dels oder Repeats

# Funktionalität der Klasse Dotplot.java

---

- a) Verarbeiten zweier Sequenzen als Input
- b) Berechnen und Speichern der Hit Matrix
- c) Ausgabe des Dotplots auf die Console
  - Mögliche Erweiterungen
    - Einlesen der Sequenzen aus Files
    - Konstruktoren Verkettung
    - Output in ein File



## Klassenvariablen und Objektvariablen

---

- Symbol zur Representierung eines Dots im Output, z.B. \*  
`private static char hitSymb;`
- Input Sequenz 1  
`private String sequence1;`
- Input Sequenz 2  
`private String sequence2;`
- Matrix zum Abspeichern der Hits: true falls Sequenzen an dieser Position das selbe Zeichen haben, sonst false  
`private boolean[][] hitMatrix;`

# Der Konstruktor

---

- Initialisierung des Dotplots mit 2 Sequenzen

```
public Dotplot(String seq1, String seq2)
```

- Füllen der Hit Matrix durch Aufruf von Berechnungs-Methode

```
private boolean[] [] computeHitMatrix(String seq1,  
String seq2)
```

## Weitere Methoden

---

- Übersetzen eines Hit Matrix Werts: Im Output soll nicht true und false stehen

```
private char getPlotChar(int pos1, int pos2)
```

- Ausgabe des Plots auf die Console: Die Sequenzen sollen als *Label* an X- und Y-Achse stehen

```
public void printPlot()
```

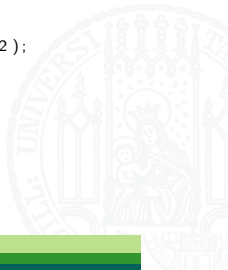
- Die main Funktion zum Starten des Programms: Initialisieren eines Dotplot Objekts und Aufrufen der printPlot() Methode

```
public static void main(String[] args)
```

# Dotplot.java

---

```
1 public class Dotplot {
2
3     private static char hitSymb = 'O'; // A symbol to represent a hit in our plot
4
5     private String sequence1; // Two input sequences
6     private String sequence2;
7     private boolean [][] hitMatrix; // Matrix to store true or false depending on hit
8
9     /**
10      * Constructor
11      */
12     public Dotplot(String seq1, String seq2) {
13         this.sequence1 = seq1;
14         this.sequence2 = seq2;
15         this.hitMatrix = computeHitMatrix(this.sequence1, this.sequence2);
16     }
17 }
```



## computeHitMatrix

---

```
1 private boolean [][] computeHitMatrix(String seq1, String seq2) {
2     // We need a two dimensional matrix to store hits
3     boolean [][] matrix = new boolean[seq1.length()][seq2.length()];
4     /*
5      * Compare every character of the two sequences with each other
6      * and save a true, if the characters are the same and false otherwise
7      */
8     for (int s1pos = 0; s1pos < seq1.length(); s1pos++) {
9         for (int s2pos = 0; s2pos < seq2.length(); s2pos++) {
10            if (seq1.charAt(s1pos) == seq2.charAt(s2pos)) {
11                matrix[s1pos][s2pos] = true;
12            }
13            else {
14                matrix[s1pos][s2pos] = false;
15            }
16        }
17    }
18    return matrix;
19 }
```



# getPlotChar

---

```
1 private char getPlotChar(int pos1, int pos2) {
2     // Translate the boolean to a character
3     if (this.hitMatrix[pos1][pos2] == true) {
4         return Dotplot.hitSymb;
5     }
6     else {
7         return ' ';
8     }
9 }
```



# printPlot

---

```
1 public void printPlot() {
2     for (int s1pos = -1; s1pos < this.sequence1.length(); s1pos++) {
3         StringBuffer nextLine = new StringBuffer();
4         for (int s2pos = -1; s2pos < this.sequence2.length(); s2pos++) {
5             if (s1pos == -1) { // First row will hold sequence 2
6                 if (s2pos == -1) {
7                     nextLine.append("-"); // In the top left corner we want a blank
8                 }
9                 else { // Otherwise, we want the characters from sequence 2 separated by a blank
10                    nextLine.append("-" + this.sequence2.charAt(s2pos));
11                }
12            }
13            else {
14                if (s2pos == -1) { // First column will hold sequence 1
15                    nextLine.append(this.sequence1.charAt(s1pos));
16                }
17                else {
18                    nextLine.append("-" + getPlotChar(s1pos, s2pos));
19                }
20            }
21        }
22        System.out.println(nextLine.toString());
23    }
24 }
```



# main

---

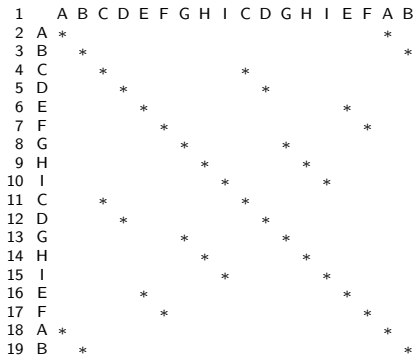
```
1 public static void main(String[] args) {
2     String s1 = "abcdefghicdghiefab".toUpperCase();
3     Dotplot dp = new Dotplot(s1, s1);
4     dp.printPlot();
5 }
```

- Initialisieren einer (oder zwei verschiedener) Input Sequenze(n)
- Erzeugen eines Dotplot Objekts (und interner Berechnung der HitMatrix)
- Ausgabe des Dotplots

# Beispiel Output

---

Dotplot für die Sequenz ABCDEFGHICDGHIEFAB gegen sich selbst:



## Erweiterung: readSequence

---

```
1 public static final String readSequence(File seq) {
2     // We use a StringBuffer instead of a string
3     StringBuffer sequence = new StringBuffer();
4     try {
5         // Convenient way to read a file
6         FileReader fr = new FileReader(seq);
7         BufferedReader sequenceReader = new BufferedReader(fr);
8         String line; // Store lines while we read through the file
9         while ((line = sequenceReader.readLine()) != null) {
10            sequence.append(line);
11        }
12        // Don't forget to close open readers!
13        fr.close();
14        sequenceReader.close();
15    } catch (FileNotFoundException fnfe) {
16        System.err.println("Cannot find file: " + seq.getPath());
17        System.exit(-1);
18    } catch (IOException ioe) {
19        System.err.println("Cannot read from file: " + seq.getPath());
20        System.exit(-1);
21    }
22    // Everything's read! Convert the StringBuffer to a String and return
23    return sequence.toString();
24 }
```



## Erweiterung: Konstruktoren Verkettung

---

```
1 public Dotplot(File seq1File, File seq2File) {  
2     this(Dotplot.readSequence(seq1File), Dotplot.readSequence(seq2File));  
3 }
```

- Konstruktor wird mit zwei Files anstatt zwei Sequenzen initialisiert
- Files werden von statischer Methode `readSequence` zu Strings eingelesen
- Der ursprüngliche Konstruktor wird mit diesen zwei Sequenz Strings aufgerufen



# Frohe Weihnachten

.. und wir sehen uns auf der Weihnachtsfeier am 21. Dezember im Gumbel!

