Rapport arbeidskrav 4

Tor-Øyvind Paulsrud Bakken

Oppgave 1 Josephus problem

4.3-5

Main-metoden:

```
public static void main(String[] args) {
    System.out.println("Den trygge plassen er: " + josephus( n: 10,  steg: 4));
}
```

Josephus:

```
public static double josephus(int n, int steg){
    Node first = new Node( e: 1);
    Node prev = first;
    for (int i = 2; i < n+1; i++) {
        Node ny = new Node(i);
        ny.forrige = prev;
        prev.neste = ny;
        prev = ny;
        first.forrige = ny;
    }
    prev.neste = first;

    Node current = first.forrige;

    //Hvis den nåverende noden linker til seg selv må det bety at det kun er igjen én person.
    while (!current.equals(current.neste)){
        //steger seg gjennom personer som er trygge før den hopper over den som blir drept.
        for (int i = 0; i < steg-1; i++) {
            current = current.neste;
        }
        current.neste = current.neste.neste;
    }
    return current.getElement();
}
```

Den første for-løkken lager nye noder og linker de sammen. Litt unødvendig laget jeg dobbeltlinket liste her da det hadde holdt med enkeltlinket.

Resultatet

```
Den trygge plassen er: 5.0
```

Som stemmer med eksempelet i boka.

Analyse
Ytterst har vi en for-løkke og en while-løkke. Noe som gir O(2n) eller bare forenklet til O(n). I tillegg er det en indre for-løkke i while-løkken som går gjennom steg-lengden, som gir O(m). Dermed blir kompleksiteten O(n)*O(m) som gir O(nm) for dette programmet.

Oppgave 2 Matche parenteser

Main-metoden

```
public static void main(String[] args) {
    String filSomSjekkes = lesFil( filnavn: "TestFile.txt");
    parantesSjekk(filSomSjekkes);
}
```

parantesSjekk

```
public static void parantesSjekk(String sjekk) {
    Stack<String> stakk = new Stack<>();
    boolean feilFunnet = false;

    for (int i = 0; i < sjekk.length(); i++) {
        String checkChar = Character.toString(sjekk.charAt(i));
        if (checkChar.matches( regex: "[({\\[]")) {
            stakk.push(checkChar);
        }
        if (checkChar.matches( regex: "[)}\\]]")) {
            switch (checkChar) {
                case ")":
                    if (!stakk.empty()) {
                        if (stakk.peek().equals("(")) {
                            stakk.pop();
                        } else {
                            System.out.println("For mange: " + stakk.peek());
                            feilFunnet = true;
                        }
                    } else {
                        System.out.println("For mange: )");
                    }
                    break;
                case "}":
                    if (!stakk.empty()) {
                        if (stakk.peek().equals("{")) {
                            stakk.pop();
                        } else {
                            System.out.println("For mange: " + stakk.peek());
                        }
```

```
                    } else {
                        System.out.println("For mange: }");
                        feilFunnet = true;
                    }
                    break;
                case "]":
                    if (!stakk.empty()) {
                        if (stakk.peek().equals("[")) {
                            stakk.pop();
                        } else {
                            System.out.println("For mange: " + stakk.peek());
                        }
                    } else {
                        System.out.println("For mange: ]");
                        feilFunnet = true;
                    }
                    break;
                default:
                    System.out.println("Something wrong: too many right-parenteces or brackets");
            }
        }
    }
    if (stakk.empty() && !feilFunnet) {
        System.out.println("Alt bra");
    } else {
        System.out.println("Mangler lukkende parentes");
    }
}
```

Metoden looper gjennom tekststrengen og finner den en venstreparentes legges denne til stakken. Hvis det er en høyreparentes så sjekker den om den passer med den parentesen som ligger øverst på stakken. Hvis de matcher fjernes den fra stakken og koden kjører videre. Hvis den finner en høyreparentes når stakken er tom printer den ut en feilmelding. Kjører hele for-løkken uten at feil er funnet sjekker den at stakken er tom. Hvis det fremdeles ligger venstreparenteser på stakken vil det si at det mangler lukkende parentes.

lesFil

```
public static String lesFil(String filnavn) {
    InputStream is = MainOppgave2.class.getResourceAsStream(filnavn);
    return new BufferedReader(new InputStreamReader(is, StandardCharsets.UTF_8)).lines().collect(Collectors.joining( delimiter: "\n"));
}
```

Resultat
Kildekode legges i TestFile. Her er resultater når jeg legger inn testene fra oppgaven

Feil: {[](()) Mangler avsluttende krøllparentes

```
Mangler lukkende parentes

Process finished with exit code 0
```

Feil: {([()])}] En klamme for mye

```
For mange: ]
```

Feil: (} Disse passer ikke sammen

```
For mange: (
Mangler lukkende parentes
```

Feil: [{] } Selv om antallene er korrekte

```
For mange: {
Mangler lukkende parentes
```

Rett men komplisert: (){{}[[]][]}{()}

```
Alt bra
```

Rett m. kode: int main(){ println("ok");}

```
Alt bra
```