Vejledende løsninger til opgave 1, 2 og 3

Opgave 1: Objektorienteret programmering

Spørgsmål 1.1

```
Animal(String n) {
    name = n;
    next = firstAnimal;
    firstAnimal = this;
}
```

Spørgsmål 1.2

```
Herbivore(String n, int g) {
    super(n);
    grassNeeded = g;
}
```

Kommentar: Bemærk at sætningen `super(n)` *skal være den første i konstruktøren.*

Spørgsmål 1.3

```
static void printAnimals() {
    for (Animal a = firstAnimal; a != null; a = a.next)
        System.out.println(a);
}
```

Spørgsmål 1.4

```
static void printHerbivores() {
    for (Animal a = firstAnimal; a != null; a = a.next)
        if (a instanceof Herbivore)
            System.out.println(a);
}
```

Spørgsmål 1.5

Sætningerne 3, 6 og 7 giver fejl under oversættelsen.

Spørgsmål 1.6

(1) Første linje i erklæringen af klassen ændres til

```
class Giraffe extends Herbivore implements Sortable {
```

(2) Klassen tilføjes en metode `lessThan`:

```
public boolean lessThan(Sortable s) {
    return neckLength < ((Giraffe) s).neckLength;
}
```

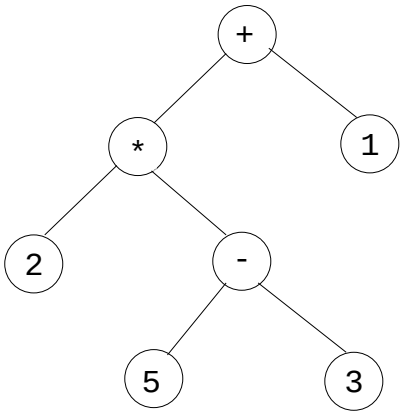
Kommentar: Bemærk at parameteren skal være af samme type som i interfacets metodespecifikation, dvs. `Sortable`. *Desuden bør metoden være erklæret som* `public`.

Opgave 2: Parsetræer

Spørgsmål 2.1

```
2 5 3 - * 1 +
```

Spørgsmål 2.2



Spørgsmål 2.3

```
class Div extends Operator {
    Div(Node l, Node r) { left = l; right = r; }
    int eval() { return left.eval() / right.eval(); }
    public String toString() { return "/"; }
}
```

Kommentar: Klassen er fremkommet ved få ændringer af en af de øvrige underklasser til `Operand`.

Spørgsmål 2.4

```
Node root = new Plus(new Mult(new Constant(2),
                              new Minus(new Constant(5),
                                        new Constant(3))),
                     new Constant(1));
```

Kommentar: En anden løsningsmulighed er følgende:

```
Constant c1 = new Constant(1);
Constant c2 = new Constant(2);
Constant c3 = new Constant(3);
Constant c5 = new Constant(5);
Node root = new Minus(c5,c3);
root = new Mult(c2,root);
root = new Plus(root,c1);
```

Spørgsmål 2.5

```
(((2*(5-3))+1)
Value = 5
```

Spørgsmål 2.6

```
Node term() {
    Node t = factor();
    while (token == MULT || token == DIV)
        if (token == MULT) {
            getToken();
            t = new Mult(t, factor());
        }
        else {
            getToken();
            t = new Div(t, factor());
        }
    return t;
}
```

Spørgsmål 2.7

```
System.out.println(new Parser().parse(“2*(5-3)+1”).eval());
```

Opgave 3: Sortering

Spørgsmål 3.1

```
a[1] = 3, a[2] = 3
```

Kommentar: `partition` *terminerer ikke, hvis midterelementet,* `a[(i+j)/2]`*, har samme værdi som et af tabelsegmentets øvrige elementer. Metoden fungerer efter hensigten, hvis tabellens elementer er forskellige.*

Spørgsmål 3.2

linje 5: `swap(a, i++, j);`

linje 8: `swap(a, i, j--);`

Kommentar: Metoden vil også være korrekt, hvis en af operatorerne `++` *eller* `--` *udelades. Men begge operatorer kan ikke udelades på samme tid.*

Spørgsmål 3.3

```
top.m = partition(a, top.i, top.j);
top.PC = 2;
top = new Frame(top.i, top.m - 1, top);
```