Lösningsförslag till tentamen för TDA540
Objektorienterad Programmering

Institutionen för Datavetenskap

CTH HT-16, TDA540

Dag: 2017-01-09, *Tid:* 14.00-18.00

Uppgift 1

a) `class` används för en klassdeklaration som är ett mall för att skapa objekt

`public` en modifierare som antyder att enheten (instansvariabel, metod, etc.) är tillgängligt för alla

`static` en modifierare som anger att en enhet (variabel eller metod) tillhör själva klassen och är gemensamt för alla klassens objekt

`void` returtyp som antyder att en metod har inget returvärde

`break` slutar exekvera nuvarande loop

`if` används för selektering mellan två satser beroende på en boolesk uttryck

`int` används för att deklarera en variabel som kan spara en heltal

`for` används för att skapa en räknareloop

b)

Klass	*Signatur*	*Returtyp*	*Statisk*
`Match`	`main(String[] args)`	`void`	ja
`Match`	`match(String pattern, String text)`	`void`	ja
`String`	`length()`	`int`	nej
`String`	`charAt(int i)`	`char`	nej
`PrintStream`	`println(int i)`	`void`	nej

c)
```
15 match("test", "kein protest, du testest");
4 for (i = 0; i <= text.length() - pattern.length(); i++)
5 for (j = 0; j < pattern.length(); j++)
6 if (pattern.charAt(0) != text.charAt(0))
```

```
7 break;
```

d) Utskriften:

```
8
17
20
```

Metoden `match` skriver ut indexen av början av förekomsterna av strängen `pattern` i strängen `text`.

e)

```
import java.util.ArrayList;
import java.util.List;

class Match {
  public static List<Integer> match(String pattern, String text) {
    List<Integer> indices = new ArrayList<>();
    int i, j;

    for (i = 0; i <= text.length() - pattern.length(); i++) {
      j = 0;
      while (j < pattern.length() && pattern.charAt(j) == text.charAt(i + j))
        j++;

      if (j == pattern.length())
        indices.add(i);
    }

    return indices;
  }

  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    System.out.print("Give pattern: ");
    String pattern = sc.nextLine();

    System.out.print("Give text: ");
    String text = sc.nextLine();

    System.out.println(match(pattern, text));
  }
}
```

Uppgift 2

- Det saknas en vänsterklämm (måsving) efter metodens parametrar.
- Variabeln `m` är av typen `int` men blir initialiserat med ett `double` värde. Det resulterar i en typfel.

- Det saknas en `return`-sats.

Förbättrade versionen:

```
public static int power(int x, int n) {
  int m = 1;
  for (int i = 0; i < n; i++)
    m *= x;
  return m;
}
```

Uppgift 3

a) Utskriften:

```
2
1
Division by zero!
Done!
```

b)

```
class Divide {
  static int[] nums   = {2, 3, 6, 8, 4, 8};
  static int[] denoms = {1, 2, 0, 2, 3};

  public static int divide(int a, int b) {
    if (a % b != 0)
      throw new ArithmeticException("Remainder not zero!");
    else
      return (a / b);
  }

  public static void main(String[] args) {
    try {
      for (int i = 0; i < nums.length; i++) {
        int number = divide(nums[i], denoms[i]);
        System.out.println(number);
      }
    } catch (ArithmeticException e) {
      System.out.println(e.getMessage());
    } catch (ArrayIndexOutOfBoundsException e) {
      System.out.println("Out of bounds index!");
    } finally {
      System.out.println("Done!");
    }
  }
}
```

c) Det är klassen `Throwable`.

Uppgift 4

```
public class Rectangle {
  private final Point ul, lr;

  public Rectangle(Point upperLeft, Point lowerRight) {
    ul = upperLeft;
    lr = lowerRight;

    if (ul.getX() >= lr.getX() || ul.getY() <= lr.getY())
      throw new IllegalArgumentException("Invalid points!");
  }

  public Point getUpperLeft() {
    return ul;
  }

  public Point getLowerRight() {
    return lr;
  }

  public int area() {
    return (lr.getX() - ul.getX()) * (ul.getY() - lr.getY());
  }

  public boolean contains(Rectangle r) {
    return (ul.getY() >= r.ul.getY() &&
            ul.getX() <= r.ul.getX() &&
            lr.getY() <= r.lr.getY() &&
            lr.getX() >= r.lr.getX());
  }

  public boolean overlaps(Rectangle r) {
    boolean res = true;

    // Do not overlap on x-axis
    if (ul.getX() > r.lr.getX() || r.ul.getX() > lr.getX())
      res = false;

    // Do not overlap on y-axis
    if (ul.getY() < r.lr.getY() || r.ul.getY() < lr.getY())
      res = false;

    return res;
  }
}
```

Uppgift 5

a) Både klass `B` och `C` är subklasser av klass `A` vilket åstadkommas med `extends A`.

b) 1. 1
2. 2
3. 1
4. 1
5. 1
6. 2

c) 1. Korrekt: 1
2. Korrekt: 2
3. Korrekt: 1
4. Inkorrekt: man får en typfel för `A` är ingen subtyp av `B`
5. Korrekt: 2
6. Inkorrekt: man får en typfeil för `C` är ingen subtyp av `B`

Uppgift 6

a)
```
class Tree  {
  Tree l;
  Tree r;
  long data;

  Tree(long data) {
    this.data = data;
  }

  Tree (long data, Tree g, Tree d) {
    this(data);
    l = g;
    r = d;
  }
}
```

b) Det stämmer, i `Match` använder vi oss bara av det imperativa delen av Java, vi har inga instansvariabler eller instansmetoder.

c)
```
import java.util.ArrayDeque;
import java.util.ArrayList;
import java.util.List;
import java.util.Queue;

public class ArityTree {
  public List<ArityTree> subtrees;
  public final long data;

  ArityTree(long data) {
```

```
    this.data = data;
    subtrees = new ArrayList<>();
  }

  public List<Long> flatten() {
    List<Long> elements = new ArrayList<>();

    elements.add(data);

    for (ArityTree tree : subtrees)
      elements.addAll(tree.flatten());

    return elements;
  }

  public List<Long> flattenBF() {
    List<Long>       elems = new ArrayList<>();
    Queue<ArityTree> queue = new ArrayDeque<>();

    queue.add(this);

    while (!queue.isEmpty()) {
      ArityTree tree = queue.remove();

      elems.add(tree.data);
      queue.addAll(tree.subtrees);
    }

    return elems;
  }
}
```

d)

```
public class Tree extends ArityTree {

  Tree(long data) {
    super(data);
  }

  Tree(long data, Tree l, Tree r) {
    this(data);
    subtrees.add(l);
    subtrees.add(r);
  }
}
```

e) se c)

f) Nej, tvärtom ja. Vi kan använda en instans av en subklass där vi förväntar oss en superklass, dvs vi kan ge ett objekt av typen `Tree` där ett objekt av `ArityTree` förväntas. Vi är ju säkert att en `Tree` har allt vad en `ArityTree` också har (`Tree` är en subtyp av `ArityTree`.