WESTERDALS Oslo ACT
EKSAMEN
PGR100 Objektorientert programmering 1

Tillatte hjelpemidler: Ingen
Varighet: 180 minutter
Dato: 15. desember 2015

5 vedlegg: Dokumentasjon og kode (side 3-8)

Denne deleksamen utgjør 75% av karakteren i PGR100. Lykke til!

Pass på at du disponerer tiden riktig i forhold til vektingen av de ulike oppgavene.

Oppgi eventuelle forutsetninger du tar.

Oppgave 1 (30%)

a) I objektorientert programmering, hva er klasser og objekter?
b) Hva gjør en konstruktør i Java?
c) Forklar begrepene kildekode og kompilator. Bruk vedlegg 2 til å konkretisere forklaringen din.
d) Se vedlegg 3. Beskriv variabelen som er deklarert i kodelinje 10 (du ser kodelinjene helt til venstre i vedlegget). Hva slags variabel er dette? Hvilken type har variabelen. Hvilket navn har den? Hvem kan aksessere den?

Oppgave 2 (10%)

En medstudent har lenge stått fast med å finne ut av hvorfor en feil dukker opp i Java-programmet sitt. Studenten forklarer:

«Jeg skjønner ikke dette. Det funket tidligere i dag! Jeg forsøker å lage en metode som skal skrive ut noen tall i en bestemt rekkefølge, men nå kommer det ikke fram noen tall i det hele tatt! Jeg har nå sittet i to timer og sett veldig nøye igjennom koden min mange ganger, men jeg kan ikke finne feilen. Java SUGER!»

Hvilke andre metoder enn å lese igjennom koden kan du anbefale studenten å benytte for å finne ut av feilen?

Oppgave 3 (5%)

Implementer (skriv) metoden `getNumberOfWords()` (se vedlegg 3, linje 32).

```
/**
 * Return the number of words in the words arraylist
 *
 * @return  the number of words in the words arraylist
 */
public int getNumberOfWords(){
    return words.size();
}
```

Oppgave 4 (10%)

Implementer (skriv) metoden `hasLuckyNumber()` (se vedlegg 4, linje 27)

```
public boolean hasLuckyNumber(int[] numbers, int luckyNumber){
    for(int i = 0; i<numbers.length ; i++){
        if(numbers[i]==luckyNumber){
            return true;
        }
    }
    return false;
}
```

Oppgave 5 (30%)

Se relevante vedlegg for denne oppgaven. Du kan anta at du har et `Example`-objekt der `words` inneholder følgende fem ord: "Katt" "Andalucia" "Lastebil" "Brus" "Eksamen" (i den rekkefølgen).

a) Hva blir skrevet ut hvis du kaller metoden `printExample` i dette `Example` -objektet?
Word starting with A:Andalucia
Big word:Lastebil

b) Forklar hvordan du har kommet fram til svaret i a).
Metoden kaller util-metoden printSomething med sitt field «words» som argument. Den arraylisten har (som oppgaven sier) de 5 ordene beskrevet i oppgaven. printSomething() itererer over disse ordene slik: Første iterasjon (Katt): Treffer ikke på stor bokstav, stor lengde eller avsluting. Andre iterasjon(Andalucia): Treffer på stor A og skriver derfor ut «Word starting with A:Andalucia». Ettersom den traff på stor A vil den ikke utføre else if(langt ord). Treffer ikke på avslutning. Tredje iterasjon(Lastebil): Treffer kun på langt ord og skriver derfor ut: «Big word:Lastebil». Fjerde iterasjon(Brus). Treffer kun på avslutning (word.endsWith("s"), noe som medfører at finished settes til true og iterasjonen (while-løkken) er ferdig.

c) Du ser kanskje at metoden `printSomething` ikke er særlig robust. Hvordan kan du kalle metoden og ende opp med å få problemer (feilmelding)?

Hvis jeg kaller metoden med et argument som ikke inneholder et ord som slutter på s, så vil jeg til slutt forsøke å aksessere en indeks som ikke finnes i arraylisten. Det vil gi meg en feilmelding (IndexOutOfBoundsException).

d) Endre metoden `printSomething` slik at den funksjonelt sett fungerer på samme måte som tidligere, men er bedre rustet mot feilmeldinger.

```
public void printSomething(ArrayList<String> words){
    if(words == null||words.size()==0){
        return;
    }
    boolean finished = false;
    int index = 0;
    String word = null;
    while(!finished && index<words.size()){
        word = words.get(index);
        if(word.startsWith("A")){
            System.out.println("Word starting with A:" + word);
        } else if(word.length()>5){
            System.out.println("Big word:" + word);
        }
        if(word.endsWith("s")){
            finished = true;
        }
        index++;
    }
}
```

Oppgave 6 (15%)

Implementer (skriv) testmetoden `testLengthOfLongestWord` (se vedlegg 5, linje 44) for å forsikre deg om at metoden `getLengthOfLongestWord` (se vedlegg 4, linje 62) fungerer som tiltenkt.

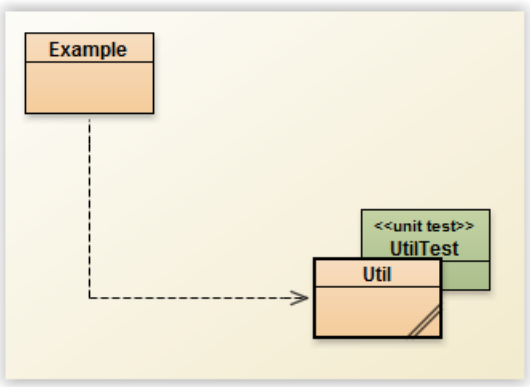
```
@Test
public void testLengthOfLongestWord()
{
    ArrayList<String> normalList = new ArrayList<>();
    normalList.add("Per");
    normalList.add("Jens");
    normalList.add("Oj");
    int result = util.getLengthOfLongestWord(normalList);
    assertEquals(result, 4);
    result = util.getLengthOfLongestWord(new ArrayList<String>());
    assertEquals(result, 0);
}
```

Hvis vi tester med en nullverdi, så vil testen avdekke at metoden ikke tar høyde for dette, men det er litt utenfor scopet til oppgaven (men fint hvis man nevner det).

- Slutt på oppgavesettet –

Vedlegg

Vedlegg 1: Oversikt over prosjektet som benyttes i denne eksamen.



Vedlegg 2: Filer i prosjektet som benyttes i denne eksamen

Navn	Dato endret	Type	Størrelse
Example.class	03.12.2015 14:03	CLASS-fil	1 kB
Example.ctxt	03.12.2015 14:03	CTXT-fil	1 kB
Example.java	03.12.2015 14:07	JAVA-fil	1 kB
package.bluej	26.11.2015 16:01	blue	1 kB
README.TXT	26.11.2015 15:25	Tekstdokument	1 kB
Util.class	03.12.2015 13:57	CLASS-fil	2 kB
Util.ctxt	03.12.2015 13:57	CTXT-fil	2 kB
Util.java	03.12.2015 14:10	JAVA-fil	2 kB
UtilTest.class	03.12.2015 13:57	CLASS-fil	1 kB
UtilTest.ctxt	03.12.2015 13:57	CTXT-fil	1 kB
UtilTest.java	03.12.2015 12:42	JAVA-fil	1 kB

Vedlegg 3: Example-koden

```java
import java.util.ArrayList;
/**
 * Example class created for exam purposes only.
 *
 * @author (Per Lauvås)
 * @version (0.1)
 */
public class Example
{
    private ArrayList<String> words;
    private Util util;

    /**
     * Constructor for objects of class Example
     */
    public Example()
    {
        words = new ArrayList<String>();
        util = new Util();
    }

    public void printExample(){
        util.printSomething(words);
    }

    /**
     * Return the number of words in the words arraylist
     *
     * @return  the number of words in the words arraylist
     */
    public int getNumberOfWords(){
        //To be implemented...
    }

    /**
     * Add a word to this objects word arraylist
     *
     * @param word  the word to be added
     */
    public void addWord(String word)
    {
        words.add(word);
    }
}
```

Vedlegg 4: Util-koden

```java
import java.util.ArrayList;
/**
 * A utility class to provide help to other classes.
 *
 * @author (Per Lauvås)
 * @version (0.1)
 */
public class Util
{

    /**
     * Constructor for objects of class Util
     */
    public Util()
    {
    }
```

```java
    /**
     * Investigates if a specific "lucky number"
     * is found in an array of integers.
     *
     * @param numbers   the numbers to investigate
     * @param luckyNumber   the number to look for
     *
     * @return true if the luckynumber is found in the array. False o
     */
    public boolean hasLuckyNumber(int[] numbers, int luckyNumber){
        //To be implemented.
    }
```

otherwise.

(koden fortsetter på neste side)

```java
/**
 * Simple example method for printing something.
 * Written for example purposes only.
 */
public void printSomething(ArrayList<String> words){
    if(words == null||words.size()==0){
        return;
    }
    boolean finished = false;
    int index = 0;
    String word = null;
    while(!finished){
        word = words.get(index);
        if(word.startsWith("A")){
            System.out.println("Word starting with A:" + word);
        } else if(word.length()>5){
            System.out.println("Big word:" + word);
        }
        if(word.endsWith("s")){
            finished = true;
        }
        index++;
    }
}
```

```java
/**
 * Retrieve the length of the longest word in the incoming array
 *
 * @param words     the words to examine
 * @return     the length of the longest word
 */
public int getLengthOfLongestWord(ArrayList<String> words)
{
    int longest = 0;
    for(String word : words){
        if(word.length()>longest){
            longest = word.length();
        }
    }
    return longest;
}
}
```

Vedlegg 5: UtilTest-koden

```
import static org.junit.Assert.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

/**
 * The test class UtilTest.
 *
 * @author  (Per Lauvås)
 * @version (0.1)
 */
public class UtilTest
{
    Util util;
    /**
     * Default constructor for test class UtilTest
     */
    public UtilTest()
    {
    }
```

(koden fortsetter på neste side)

```
22      /**
23       * Sets up the test fixture.
24       *
25       * Called before every test case method.
26       */
27      @Before
28      public void setUp()
29      {
30          util = new Util();
31      }
32
33      /**
34       * Tears down the test fixture.
35       *
36       * Called after every test case method.
37       */
38      @After
39      public void tearDown()
40      {
41      }
42
```

```
43      @Test
44      public void testLengthOfLongestWord()
45      {
46          //To be implemented
47      }
48  }
```