Object Oriented Programming TDA540
Object-Oriented Programming
(Objektorienterad Programmering)

Day: 2020-01-18, Time: 14.00-18.00

Examinator: Contact:	Krasimir Angelov 031 772 10 19
Result:	will be available via Ladok
Grade boundaries:	3:a 24 points 4:a 36 points 5:a 48 points max 60 points
Numbers in brackets:	Maximal number of points for each question
Allowed material:	Cay Horstmann: Java for everyone (2nd edition) or Jan Skansholm: Java direkt med Swing. Markings and small footnotes are allowed. You can also use an English-Swedish dictionary if needed.
Please take note:	Write clearly and make sure to hand in your solutions in the right place. Start each question on a new piece of paper. Do not write on the back side of the paper.
Additional remarks:	The exam consists of 6 questions, not ordered by difficulty. All programs should be structured and easy to read. Don't forget to use indentation! For the correction of program code, logical flaws are considered more serious than minor syntax errors.

Question 1 (17 points)

Take a look at the program below:

```
1  public class Hotel {
2      private Clock c;
3      private int checkInHour;
4      private int checkOutHour;
5
6      public Hotel(int checkInHour, int checkOutHour) {
7          this.checkInHour  = checkInHour;
8          this.checkOutHour = checkOutHour;
9          this.c = new Clock();
10     }
11
12     public void checkIn(Guest guest, int days) {
13         if (c.getCurrentHour() < checkInHour) {
14             throw new RuntimeException("Sorry, checking in is possible"+
15                                        " only after "+checkInHour+":00");
16         }
17         guest.message("Welcome");
18         guest.setCheckoutDate(c.getCurrentDate()+days);
19     }
20
21     public void checkOut(Guest guest) {
22         int extra = c.getCurrentDate() - guest.getCheckoutDate();
23         if (c.getCurrentHour() > checkOutHour)
24             extra++;
25         if (extra > 0)
26             guest.message("You will be charged"+
27                           " for "+extra+" extra nights");
28         guest.message("Have a nice day");
29     }
30 }
31
32 public class Guest {
33     private int checkOutDate;
34
35     public void message(String msg) {
36         System.out.println(msg);
37     }
38
39     public void setCheckoutDate(int date) { checkOutDate = date; }
40     public int getCheckoutDate() { return date; }
41 }
```

a. List all method calls in this program (not method declarations). For each method call, give:

- The line number(s)
- The return type of the method
- The formal and the actual parameters of the method call

(5 points)

Answer:

Line #	Method	Returns	Formal	Actual
13	getCurrentHour	int*	none	none
17	message	void	msg	"Welcome"
18	setCheckoutDate	void	date	c.getCurrentDate() + days
22	getCurrentDate	int	none	none
22	getCheckoutDate	int	none	none
23	getCurrentHour	int	none	none
26	message	void	msg	"You will be ..."
28	message	void	msg	"Have a nice day"
36	println	void	-	msg

Notes:

- The class `Clock` is not included in the listing. However, from the syntax you can infer that `getCurrentHour` and `getCurrentDate` have no arguments. `getCurrentDate` returns integer because on line 22 it is used for arithmetics with `getCheckoutDate` and the result of the substraction is of type `int` too. The result from method `getCurrentHour` is compared with another integer, i.e. checkOutHour, on line 23 and therefore it must be of numeric type. Strictly speaking `getCurrentHour` can return any numeric type, but *int* is the most logical choice.
- On lines 9 and 14 there are two objects created. The object creation implicitly calls the corresponding constructor. Whether a constructor is counted as method or not is up to interpretation. No points were discounted if those two lines were not listed.

b. Line 40 actually contains an error. Write the corrected code and explain whether this is a compile-time or runtime error.

Answer:

Variable `date` is not defined. The only logical solution is to replace it with `checkOutDate`. (4 points)

c. What text is printed when we run the following code on 18 January:

```
1 Guest alice = new Guest();
2 Hotel hotel = new Hotel(15,12);
3 hotel.checkIn(alice,2);
```

and then the following on 21 January at 16.00:

```
1 hotel.checkOut(alice);
```

Here we assume that line 40 is corrected.

Answer: The question doesn't say at what time on 18 January the code was executed. If the code is executed before 15:00, then the program prints just raises an exception. If the code is executed after that then the output is:

```
Welcome
```

and then on 21 January:

```
You will be charged for 2 extra nights
Have a nice day
```

(4 points)

d. Look at methods `message` and `setCheckoutDate`. Which of these two can be made into a static method. Motivate both why the method could be static and why the other cannot be.

Answer:

Method `message` can be made static because it does not refer to any instance variables. On the contrary `setCheckoutDate` cannot.

(4 points)

Question 2 (5 points)

The Goldbach Conjecture is a yet unproven conjecture stating that every **even integer** greater than two is the sum of **two prime numbers**. The conjecture has been tested up to 400,000,000,000,000 but the following code is our very naïve attempt for testing:

```
1  boolean possible = false;
2  for (int n = 4; n < Integer.MAX_VALUE; n++) {
3      for (int i = 2; i < n; i++) {
4          if (isPrime(i && n-i)) {
5              possible = true;
6              System.out.println(i+(n-i)=n);
7              break;
8          }
9      }
10     if (!possible) {
11         System.Out.println(n+" is a counterexample");
12         break;
13     }
14 }
```

However, the program contains 5 errors. For each error, give the line number and whether it is a syntactic or a logical error.

Answer: The correct program should be:

```
1  for (int n = 4; n < Integer.MAX_VALUE; n += 2) {  // 1. here we skip odd numbers
2                                                    // (logical error)
3      boolean possible = false;                     // 2. reset the variable to false
4                                                    // (logical error)
5      for (int i = 2; i < n; i++) {
6          if (isPrime(i) && isPrime(n-i)) {          // 3. was syntactic error
7              possible = true;
8              System.out.println(i+"+"+(n-i)+"="+n); // 4. was syntactic error
9              break;
10         }
11     }
12     if (!possible) {
13         System.out.println(n+" is a counterexample");  // 5. Out -> out
14         break;
15     }
16 }
```

Question 3 (6 points)

What is printed out when we run the main method of the following Java program?

```
1  public class Person {
2      private String name;
3
4      public Person(String name) {
5          this.name = name;
6      }
7
8      public boolean equals(Person other) {
9          return this.name == other.name;
10     }
11
12     public static void main(String[] args) {
13         Person alice = new Person("Alice");
14         Person kate  = alice;
15
16         System.out.println(alice==kate);
17         System.out.println(alice.equals(kate));
18
19         kate.name = "Kate";
20
21         System.out.println(alice==kate);
22         System.out.println(alice.equals(kate));
23
24         kate       = new Person(kate.name);
25         kate.name = "Alice";
26
27         System.out.println(alice==kate);
28         System.out.println(alice.equals(kate));
29     }
30 }
```

Answer:

```
true
true
true
true
false
false
```

Question 4 (16 points)

The exponential function can be approximated as the sum:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots$$

i.e. each of the terms has the form $\frac{x^n}{n!}$ and in principle the summation can continue infinitely. As usual $n!$ denotes factorial and is the product:

$$n! = 1 \cdot 2 \cdot 3 \dots n$$

Write a static method:

```
public static double exp(double x, int n)
```

which computes e^x by computing the above sum up to the n-th term.

Note: In the implementation you are not allowed to use any of the standard methods in the `java.lang.Math` class.

Answer:

```
1 public static double exp(double x, int n) {
2     double sum = 0;
3     double factor = 1;
4     for (int i = 1; i <= n; i++) {
5         sum += factor;
6         factor *= x/i;
7     }
8     return sum;
9 }
```

Question 5 (8 points)

Implement the classes Movie and Cinema:

- Each Movie is represented by a title, length and genre. The titles and the genres are strings while the length is measured in hours and is represented with a double. The class should have a constructor which initializes all attributes. For each attribute there must be a getter.

- Each Cinema contains a list of movies which are shown at the moment. Implement a method for adding a new movie. There should be also a method which prints the list of movies. Each movie must be on a new line. For every movie print the title, the genre and the length.

Answer:

```
1  public class Movie {
2      private String name;
3      private String genre;
4      private double length;
5
6      public Movie(String name, String genre, double length) {
7          this.name   = name;
8          this.genre  = genre;
9          this.length = length;
10     }
11
12     public String getName()   { return name;   }
13     public String getGenre()  { return genre;  }
14     public double getLength() { return length; }
15 }
16
17 import java.util.*;
18
19 public class Cinema {
20     private List<Movie> movies;
21
22     public Cinema() {
23         this.movies = new ArrayList<Movie>();
24     }
25
26     public void add(Movie movie) {
27         movies.add(movie);
28     }
29
30     public void print() {
31         for (Movie m : movies) {
32             System.out.println(m.getName()+" "+m.getGenre()+" "+m.getLength());
33         }
34     }
35 }
```

Question 6 (8 points)

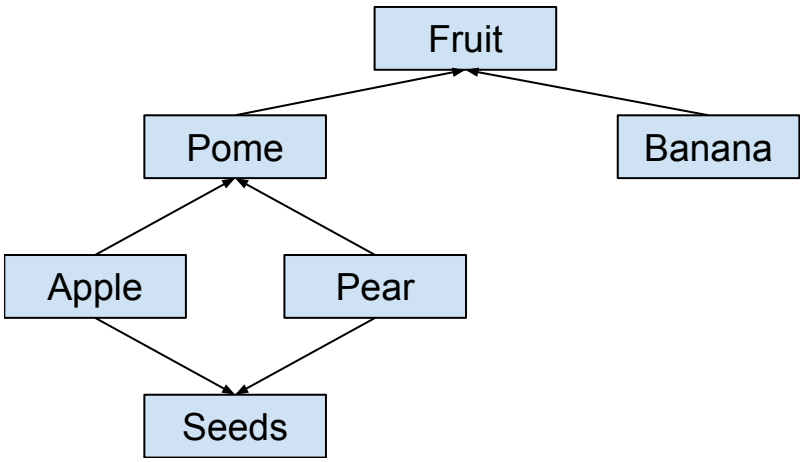
Take a look at the following interfaces and classes that model various kinds of fruits.

```
1  public class Fruit {
2  }
3  public interface Seeds {
4      public int nrOfSeeds();
5  }
6  public class Pome extends Fruit {
7  }
8  public class Apple extends Pome implements Seeds {
9      public int nrOfSeeds() { ... }
```

```
10  }
11  public class Pear extends Pome implements Seeds {
12      public int nrOfSeeds() { ... }
13  }
14  public class Banana extends Fruit {
15  }
```

- Draw a class diagram where each type X in this hierarchy is represented by a node, and draw an arrow from X to Y if Y is a direct supertype of X. (3 points)
- The main method below contains five type errors. Indicate the line number of each incorrect statement and explain in one sentence why it is wrong. (5 points)

```
1  public static void main(String[] args) {
2      Fruit fruit;
3      Banana banana;
4      Pome pome;
5      Pear pear;
6      Apple apple;
7
8      fruit = new Apple();
9      banana = new Banana();
10     pome = banana;
11     banana = (Banana) new Fruit();
12     pome = (Pome) fruit;
13     pear = pome;
14     apple = fruit;
15     System.out.println(((Seeds) banana).nrOfSeeds());
16 }
```



- Line 10: `Pome` is not subclass of `Banana`
- Line 11: `Fruit` is not subclass of `Banana` and type cast is impossible.
- *Line 12: Note: there is no error here. Type cast is possible.*
- Line 13: `Pome` is not subclass of `Pear` and type cast is impossible.
- Line 14: `Fruit` is not subclass of `Apple` and type cast is impossible. Type cast is needed.
- Line 15: `Banana` does not implement `Seeds`.