Preparation for First Session

Opgave 7.2.1

How do you declare an array reference variable and how do you create an array?

Besvarelse

So whenever we declare an array, we first of all start by writing the primitive datatype with squared paranthes. And then we write the array name and use the curly brackets to write the numbers inside.

Example:

```
Public class Eksempel{

   Public static void main(String[] args){

    int[] list {1,2,3,4,5,6,7,8,9,10}

   }

}
```

Opgave 7.2.2

When is the memory allocated for an array?

Besvarelse

In this case we can see that the memory for the array is located inside the given curly brackets list.

But we need to keep in mind, and that once a array is defined then we cant change the length of the array during the type casting.

Opgave 7.2.3

What is the output of the following code?

Besvarelse

So, we have written the code inside IntelliJ.

```java
public class Main {
    public static void main(String[] args) {
        int x = 30;
        int[] numbers = new int[x];
        x = 60;
        System.out.println("x is " + x);
        System.out.println("The size of numbers is " + numbers.length);
    }
}
```

```
C:\Users\vivek\.jdks\openjdk-18.0.
x is 60
The size of numbers is 30

Process finished with exit code 0
```

Opgave 7.2.4

Indicate true or false for the following statements:

a. Every element in an array has the same type.
- That is true, because once the primitive datatype is defined for the array, then the index-list is specified with the datatype.

b. The array size is fixed after an array reference variable is declared.
- That is true, because that is the characteristics of an array that once the array is declared it cannot be changed. Whereas an interface such as an arraylist, which is a modified version of the array, it can be changed throughout the code lines with methods.

c. The array size is fixed after it is created.
- That is true, and we can use the same reason has mention in part b.

d. The elements in an array must be of a primitive data type.
- That is true, it is very important.

Opgave 7.2.5

Which of the following statements are valid?

a. Int i = new int(30);
- This is valid.

b. Double d[] = new double[30];
- This is not valid.

c. Char[] r = new char(1….30);
- This is not valid.

d. Int i[] = (3,4,3,2);
- This is not valid.

e. Float f[] = {2,3,4,5,6,6};
- This is not valid.

f. Char[] c = new char();
- This is not valid.

Opgave 7.2.6

How do you access elements in an array?

Besvarelse

The way we access an element inside an array, is by writing the name of the array and squared brackets and thereafter we write the index number of which place we want to target.

Opgave 7.2.7

What is the array index type? What is the lowest index? What is the representation of the third element in an array named a?

Besvarelse

The array index type is the placement of the different values inside the list of the array. In this case we {0,1,2,3,4,5} are the values which are placed inside, and I have placed them in the queue which they can be accessed on. In this case we can see that place 0 is the lowest index and the third element inside a array named a would most likely not be anything because we have not got a descriptions. But for fun sake it could have been c 😊

Opgave 7.2.8

Write statement to do the following:

a. Create an array to hold 10 double values.
b. Assign the value 5.5 to the last element in the array.
c. Display the sum of the first two elements.
d. Write a loop that computes the sum of all elements in the array.
e. Write a loop that finds the minimum element in the array.
f. Randomly generate an index and display the element of this index in the array.
g. Use an array initializer to create another array with the initial values 3,5,5,5,4.52, and 5.6.

Besvarelse

Looking at the code below, we can see the following solution to task 7.2.8.

```
public class Main {
    public static void main(String[] args) {
        double[] list = {0.0,1.1,1.2,2.3,2.4,3.5,3.6,4.7,4.8,5.5};
        System.out.println(list[0]+list[1]);
        double sum = 0.0;
        for(int i=0; i<list.length;i++){
            sum = sum + list[i];
        }
        System.out.println("The sum of evey element combined is: " + sum);
        for(int i=0; i<list.length;i++){
            System.out.println(list[0]);
        }
    }
}
```

```
double[] liste = {3.5,5.5,4.52,5.6};
System.out.println(liste);
```

```
C:\Users\vivek\.jdks\openjdk-18.0.2.1\bin'
1.1
The sum of evey element combined is: 29.1
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
```

Opgave 7.2.9

What happens when your program attempts to access an array element with an array element with an invalid index?

Besvarelse

The console shows error, because if you are trying to access any value outside the defined index-row you won't be able to get any value on the console.

Opgave 7.2.10

Identify and fix the errors in the following code:

Besvarelse

X

Opgave 7.2.11

What is the output of the following code?

Besvarelse

We can in this case see that we have got the following results.

```
public class Main {
    public static void main(String[] args) {
        int list[] = {1,2,3,4,5,6};
        for (int i = 1; i<list.length;i++){
            list[i] = list[i-1];
        }
        for(int i = 0; i < list.length; i++){
            System.out.println(list[i] + " ");
        }
    }
}
```

```
C:\Us
1
1
1
1
1
1
```

Opgave 7.5.2

Once an array is created, its size cannot be changed. Does the following code resize the array?

Besvarelse

Yes, that is correct and that is because we can see here that we have updated the length from 10 to 20.

Opgave 11.2.1

True or false? A subclass is a subset of a superclass

Besvarelse

Yes, that is true a subclass is always a subset of a superclass which means that it is a class under an upper class in the hierarchy and has inherited attributes and methods from the upper class.

Opgave 11.2.2

What keyword do you use to define a subclass?

Besvarelse

In this case when we are using polymorphism and we are trying to reference to the most upper class in the hierarchy, then we use the keyword “super”. Forexample like super.methodname();

Opgave 11.2.3

What is a single inheritance? What is multiple inheritance? Does Java support multiple inheritance?

Besvarelse

In this case we can see that single inheritance is where we use the “extends” keyword to inherit attributes and methods from one class to another. But when we talk about multiple inheritance, then we can see that the Java does not support multiple inheritance though it is possible to make multiple inheritance through interfaces.

Opgave 11.3.1

What is the output of running the class C in (a)? What problem arises in compiling the program in (b)?

Besvarelse

We can in this case see, that the result from our code would not be anything, because we are instantiating an object and not writing a reference to a method from the object.

Opgave 11.3.2

How does a subclass invoke its superclass’s constructor?

Besvarelse

The way a constructor may invoke an overloaded constructor, or its superclass constructor is if it is invoked explicitly. Looking at the page number 441, we can see the example where we have two constructors with the same name, but one constructor with one parameter. In this case we can see that the Employee-constructor in the extending part is envoked because it is inheriting from Person and that means that the Employee is a subclass of Employee and is printing from Person.

But along with that we can see that because we have another constructor with the same class-name then it is invoked because we are inheriting from another constructor.

Opgave 11.3.3

True or false? When invoking a constructor from a subclass, its superclass's no-arg constructor is always invoked.

Besvarelse

Not always, it is only if the subclass constructor is not defined explicitly. Forexample we saw that the apple-class constructor was defined indirectly and because Apple is a subclass of Fruit, then we can see that the superclass no-args constructor is invoked.

Opgave 11.4.1

True or false? You can override a private method defined in a superclass.

Besvarelse

You cannot override a private or static method in java. But if it is an abstract class, then you are able override through inheritance and polymorphism.

Opgave 11.4.2

True or false? You can override a static method defined in a superclass.

Besvarelse

No, that is the same concept as mentioned before. You cannot override private and static methods in java.

Opgave 11.4.3

How do you explicitly invoke a superclass's constructor from a subclass?

Besvarelse

The way you can invoke a superclass's constructor from a subclass is by using the "super" keyword in the classname of the constructor and then you are able to access the superclass constructor from the subclass.

Opgave 11.4.4

How do you invoke an overridden superclass method from a subclass?

Besvarelse

It is not, that possibly thought you can change the parameters and method definition of the superclass and then you are able to indirectly able to invoke the subclass methods.

Opgave 11.5.1

Identify the problems in the following code:

Besvarelse

Looking at the code given from the text, we have made the following changes with the help of IntelliJ.

Just to notify, we have corrected code line 20-23 and have replaced with the constructor of B as it created an error in the original code.

```java
public class Circle {
    3 usages
    private double radius;

    1 usage
    public Circle(double radius){
        radius = radius;
    }
    public double getRadius(){
        return radius;
    }
    1 usage  1 override
    public double getArea(){
        return radius*radius*Math.PI;
    }
}
```

```java
public class B extends Circle{
    1 usage
    private double length;

    public B(double radius) {
        super(radius);
    }

    1 usage
    @Override
    public double getArea(){
        return getArea() * length;
    }
}
```

Opgave 11.5.2

Explain the difference between method overloading and method overriding.

Besvarelse

Method Overloading: This is a feature, which gives the class the access to have more than one method with the same name, but with different parameters.

Method Overriding: Is where we use polymorphism. This means, that we have a method, but we use it in different ways.

Opgave 11.5.3

If a method in a subclass has the same signature as a method in its superclass with the same return type, is the method overridden or overloaded?

Besvarelse

The method is neither overridden nor overloaded. It is just written in the same way, as it is described in the superclass.

Opgave 11.5.4

If a method in a subclass has the same signature as a method in its superclass with a different return type, will this be a problem?

Besvarelse

This will be considered as polymorphism, because we have the same method but we are using it in many different forms and ways.

Opgave 11.5.5

If a method in a subclass has the same name as a method in its superclass with different parameter types, is the method overridden or overloaded?

Besvarelse

As mentioned before, that when one class has access to one method with different parameters then it is considered to be overloading.

Opgave 11.5.6

What is the benefit of using the @Override annotation?

Besvarelse

The benefit of using @Override-annotation is that the system is notified that we are using the same method different times in the polymorphism context.

Opgave 11.7.1

What are the three pillars of objectoriented programming? What is polymorphism?

Besvarelse

In this case we can see the three pillars of objectoriented programming is encapsulation, inheritance and polymorphism. Whereas polymorphism means different forms and that means that we are overriding or using one same method in the same manner, just in different contexts in different classes.

Opgave 11.8.1

What is dynamic binding?

Besvarelse

According to google and the text, it can be understood as the dynamic binding uses objects to resolve to bind.

Opgave 11.8.2

Describe the difference between method matching and method binding.

Besvarelse

In this case we can see that whenever we match a method, then we check to see if the datatype which is written in the method is matching the way the method is being expressed.

Whereas Method binding is where we use association through method call to other classes methods and use them for implementation.

Opgave 11.8.3

Can you assign new int[50], new Integer[50], new String[50], or new Object[50] into a variable of Object[] type?

Besvarelse

It is too risky to say, because it depends on how you want to declare the variable. If you are talking about an array, then it would be most likely possible, but for variables in this case it would be not possible because the way we are definition our variable is not “codefully-right”.

Opgave 11.8.4

What is wrong in the following code?

Besvarelse

In this case we can see that the following code, which we have corrected. It is important to notify, that we have imported a static package. But we can also see here, that we have corrected int[] to Integer[].

```
import static com.sun.org.apache.bcel.internal.classfile.Utility.printArray;

public class Main {
    public static double main(String[] args) {
        Integer[] list1 = {12,24,55,1};
        Double[] list2 = {12.4,24.0,55.2,1.0};
        Integer[] list3 = {1,2,3};
        printArray(list1);
        printArray(list2);
        printArray(list3);
    }
    3 usages
    public static void printArray(Object[] list){
        for (Object o: list)
            System.out.println(o + " ");
        System.out.println();
    }
}
```

Opgave 11.8.5

Show the output of the following code:

Besvarelse

We have written the code in IntelliJ and have got the following results for part a.

```
import static com.sun.org.apache.bcel.intern

public class Main {
    public static void main(String[] args) {
        new Person().printPerson();
        new Student().printPerson();
    }
}
```

```
1 usage
public class Student extends Person{
    1 usage
    @Override
    public String getInfo(){
        return "Student";
    }
}
```

```
2 usages   1 inheritor
public class Person {
    1 usage   1 override
    public String getInfo(){
        return "Person";
    }
    2 usages
    public void printPerson(){
        System.out.println(getInfo());
    }
}
```

```
C:\Users\
Person
Student
```

We have written the code in IntelliJ and have got the following results for part b.

```
import static com.sun.org.apache.bcel.intern

public class Main {
    public static void main(String[] args) {
        new Person().printPerson();
        new Student().printPerson();
    }
}
```

```
1 usage
public class Student extends Person{
    @Override
    private String getInfo(){
        return "Student";
    }
}
```

```
2 usages   1 inheritor
public class Person {
    1 usage   1 override   1 related problem
    public String getInfo(){
        return "Person";
    }
    2 related problems
    private void printPerson(){
        System.out.println(getInfo());
    }
}
```

```
\Users\vivek\Downloads\vimis22-main\MiniFirstPreparation\src\Student.java:3:19
va: getInfo() in Student cannot override getInfo() in Person
attempting to assign weaker access privileges; was public
```

Opgave 11.8.6

Show the output of following program.

Besvarelse

The following code, shows how we have solved it in IntelliJ.

```
import static com.sun.org.apache.bcel.intern

public class Test {
    public static void main(String[] args) {
        A a = new A( t: 3);
    }
}
```

```
2 usages                                    ⚠ 1
public class A extends B{
    1 usage
    public A(int t){
        System.out.println("A's constructor is invoked");
    }

}
```

```
1 usage   1 inheritor
public class B {
    1 usage
    public B(){
        System.out.println("B's constructor is invoked");
    }
}
```

We get the following result in the console and therefore we can say that constructor of Object is not invoked when new A(3) is invoked. (Hope it is the right answer).

```
C:\Users\vivek\.jdks\openjdk-18.0.2
B's constructor is invoked
A's constructor is invoked

Process finished with exit code 0
```

Opgave 11.8.7

Show the output of following program:

Besvarelse

We have written the code in IntelliJ, and we can say that we get the following result:

```
import static com.sun.org.apache.bcel.intern

public class Test {
    public static void main(String[] args) {
        new A();
        new B();
    }
}
```

```
C:\Users\vivek\.jdks\openjdk-18.0.2

Process finished with exit code 0
```

Opgave 13.2.1

Which of the following classes defines a legal abstract class?

```
class A {
  abstract void unfinished() {
  }
}
```
(a)

```
public class abstract A {
  abstract void unfinished();
}
```
(b)

```
class A {
  abstract void unfinished();
}
```
(c)

```
abstract class A {
  protected void unfinished();
}
```
(d)

```
abstract class A {
  abstract void unfinished();
}
```
(e)

```
abstract class A {
  abstract int unfinished();
}
```
(f)

Besvarelse

We can see, that a, b, c, e, f are known as legal abstract classes because we can clearly see that only the abstract method is applicable in the abstract class.

Opgave 13.2.2

The getArea() and getPerimeter() methods may be removed from the GeometriObject class. What are the benefits of defining getArea() and getPerimeter() as abstract methods in the GeometricObject class?

Besvarelse

The benefits of defining a getArea() and getPerimeter() is that other methods cant override or access these methods. That means, that if there is something defined inside the methods, then it cant be overridden by inheritance.

Opgave 13.2.3

True or false?

a. An abstract class can be used just like a nonabstract class except that you cannot use the new operator to create an instance from the abstract class.
b. An abstract class can be extended.
c. A subclass of a nonabstract superclass cannot be abstract.
d. A subclass cannot override a concrete method in a superclass to define it as abstract.
e. An abstract method must be nonstatic.

Besvarelse

A. TRUE: We can see that the abstract can not be directly used to create an instance of a class, but it can through polymorphism be used to create an object.
B. TRUE: Yes, it can be. The abstract class can be extended to other classes.
C. TRUE: You cannot make a subclass into a abstract class, if it is already nonabstract.

D. FALSE: A subclass is able to override a method from the superclass and define it as abstract.
E. FALSE: A abstract method doesn't need to be nonstatic.

Opgave 13.3.1

Why do the following two lines of code compile but cause a runtime error?

```
Number numberRef = new Integer(0);
Double doubleRef = (Double)numberRef;
```

Besvarelse

We can in this case see, that we are using type casting out of the primitive datatypes. In this case we can see that we are converting a number from a datatype named Number and thereafter we are trying to convert it into a double. Therefore, we are getting a runtime error.

Plus, our datatype shows that the new Integer does not match with the datatype declared before the variable.

Opgave 13.3.2

Why do the following two lines of code compile but cause a runtime error?

```
Number[] numberArray = new Integer[2];
numberArray[0] = new Double(1.5);
```

Besvarelse

We can see clearly that the length of the array has been defined to 0, which means that there is no array.

Then we can see that we are trying to convert this array into a double and trying to put a length, which is not possible in simple arrays. Because of these reasons, there is a runtime error.

Opgave 13.3.3

Show the output of the following code:

```
public class Test {
  public static void main(String[] args) {
    Number x = 3;
    System.out.println(x.intValue());
    System.out.println(x.doubleValue());
  }
}
```

Besvarelse

We can in this case see that intValue converts the value into 3 and doubleValue converts the value into 3.0.

```java
public class Main {
    public static void main(String[] args) {
        Number x = 3;
        System.out.println(x.intValue());
        System.out.println(x.doubleValue());
    }
}
```

```
C:\Users\vive
3
3.0
```

Opgave 13.3.4

What is wrong in the following code? (Note the compareTo method for the Integer and Double classes was introduced in Section 10.7.)

```java
public class Test {
  public static void main(String[] args) {
    Number x = new Integer(3);
    System.out.println(x.intValue());
    System.out.println(x.compareTo(new Integer(4)));
  }
}
```

Besvarelse

You can see that IntelliJ shows the following result of the code compiling.

```java
public class Main {
    public static void main(String[] args) {
        Number x = 3;
        System.out.println(x.intValue());
        System.out.println(x.compareTo(new ~~Integer~~( value: 4)));
    }
}
```

Our console shows the following output.

```
C:\Users\vivek\Downloads\vimis22-main\FileWritinc
java: cannot find symbol
  symbol:   method compareTo(java.lang.Integer)
  location: variable x of type java.lang.Number
```

Opgave 13.3.5

What is wrong in the following code?

```java
public class Test {
  public static void main(String[] args) {
    Number x = new Integer(3);
    System.out.println(x.intValue());
    System.out.println((Integer)x.compareTo(new Integer(4)));
  }
}
```

Besvarelse

In this case we can see the following output.

```java
public class Main {
    public static void main(String[] args) {
        Number x = new Integer( value: 3);
        System.out.println(x.intValue());
        System.out.println((Integer)x.compareTo(new Integer( value: 4)));
    }
}
```

```
C:\Users\vivek\Downloads\vimis22-main\FileWritin
java: cannot find symbol
  symbol:   method compareTo(java.lang.Integer)
  location: variable x of type java.lang.Number
```

Opgave 13.4.1

Can you create a Calendar object using the Calendar class?

Besvarelse

The following example above, shows the creation of a GregorianCalender.

But this is only possible if the main-method is inside the class, so therefore we can conclude that if the Calendar Class contains a main-method, then it is possible to create a Calendar Object.

Opgave 13.4.2

Which method in the Calendar class is abstract?

Besvarelse

X

Opgave 13.4.3

How do you create a Calendar object for the current time?

Besvarelse

A possible way to create a object with the current time inside, is to create a method named Date() where you can through inheritance use the java.util.Date and include the current Date and time.

Opgave 13.4.4

For at Calendar object c, how do you get its year, month, data, hour, minute, and second?

Besvarelse

You have to use the java.util.Date package which you can import in IntelliJ.

Thereafter you will be able to get different methods, and thereafter you can include them in your method.

Then you can use inheritance and polymorphism and get access to the object under the main method.