Introduction to Programming

Written Exam 2025

Problem 1. Explain the difference between print and println.

Problem 2. Explain the difference between compile-time errors and runtime errors.

Problem 3. Explain the role of the Java compiler.

Problem 4. Assume you have written a program `Sum.java` that takes two integers and prints their sum. What command(s) can be used to compile and run the program?

Problem 5. Assume you only have access to the Java compiler, how would you determine if `strictfp` is a reserved keyword?

> **Use elementary imperative programming language constructs, including: primitive data types, local variables, assignment, arrays, if-then-else, and for- and while loops.**

Problem 6. The following program is intended to print the command-line arguments in reverse order. Fill in the missing part. (Don't repeat the whole program. Just write the missing part.)

```
public class ReverseArgs {
    public static void main(String[] args) {
        for (/* MISSING */; i >= 0; i--) {
            System.out.println(args[i]);
        }
    }
}
```

Problem 7. The program below is intended to print the number of adjacent elements that are equal. For example:

```
$ java CountAdjacent hello hello world world world java world
hello: 2
world: 3
java: 1
world: 1
```

Fill in the missing parts of the program (don't repeat the whole program.):

```
public class CountAdjacent {
    public static void main(String[] args) {
        if (args.length == 0) {
            return;
        }

        String current = args[0];
        int count = 1;

        for (int i = 1; i < args.length; i++) {
            if (/* MISSING A */) {
                count++;
            } else {
                /* MISSING B */
                current = args[i];
                count = 1;
            }
        }

        System.out.println(current + ": " + count);
    }
}
```

Problem 8. What is a constructor?

Problem 9. Complete the following method:

```
public static Map<String, List<String>> getChildren() {
    // Construct a list with the strings "Bart" and "Lisa".
    // Construct a map with two keys "Homer" and "Marge" that
    // are associated with the list.
    // Return the map.
}
```

You may assume that `java.util.*` is imported.

Problem 10. When should you use a `while`-loop and when should you use a `for`-loop?

Problem 11. Write a complete Java program that takes a positive integer n and then prints a triangle with height n and width n as follows:

```
$ java Diag 3
..*
.*
*
```

```
$ java Diag 5
....*
...*
..*
.*
*
```

Problem 12. What happens when you run the following program?

```
public class Main {
    public static void main(String[] args) {
        String[] a = new String[100];
        System.out.println(a[99]);
        System.out.println(a[100]);
    }
}
```

Problem 13. Write a complete program that when run causes a `StackOverflowException`.

Problem 14. What are the essential parts of a recursive algorithm?

Problem 15. What does "step-over" do in the debugger?

Problem 16. What is a stack frame?

Problem 17. When must a method contain a return statement?

Problem 18. What happens when you run the program:

```
public static void main(String[] args) {
    System.out.println(1.0 / 0.0);
}
```

Problem 19. What happens when you run the program:

```
public static void main(String[] args) {
    int[] a = {1, 2, 3};
    System.out.println(a);
}
```

Problem 20. What happens when you run the program:

```
public static void main(String[] args) {
    int x = 1;
    double y = 1.0;
    String z = "3.0";
    System.out.println(x + y + z);
}
```

Problem 21. Write a minimal class `Lemon` that is both a `Fruit` and a `Product`:

```
public interface Fruit {
    String getTaste(); /* sweet, sour, ... */
}
public interface Product {
    int getPrice(); /* 5 DKK, ... */
}
```

Problem 22. Rewrite the class below to use generics, i.e. it should be possible to have a pair of an Integer and String, but also a pair of a Lemon and a Lemon:

```
class Pair {
    private Integer v1;
    private String v2;

    public Pair(Integer v1, String v2) {
        this.v1 = v1;
        this.v2 = v2;
    }

    public Integer getV1() { return v1; }

    public String getV2() { return v2; }
}
```