Programming with Java Module 1

Variables and Data Types

Independent Part

Contents

1 Calculation of the Braking Distance 3
   1.1 Introduction .................................................. 3
   1.2 Task and program requirements .............................. 3

2 Compound interest calculation 3
   2.1 Introduction .................................................. 3
   2.2 Problem definition and program requirements ............... 3
   2.3 Extension .................................................. 4

3 ATM (cash machine) 4
   3.1 Introduction .................................................. 4
   3.2 Problem definition ........................................... 4
   3.3 Intermediate steps ........................................... 4
   3.4 Extension .................................................. 5
The authors try the best to provide an error free work, however there still might be some errors. The Authors are thankful for your feedback and suggestions.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

http://creativecommons.org/licenses/by-nc-nd/4.0/
1 Calculation of the Braking Distance

1.1 Introduction

The overall stopping distance describes the distance that a vehicle covers from the time of the appearance of an obstacle until the stanstill. The stopping distance is composed of the reaction and the braking distance:

\[ \text{stopping distance} = \text{reaction distance} + \text{braking distance} \]

In simplified terms, the reaction and the braking distance can be calculated by the following formula (reaction and braking distance in meters; speed in km/h).

\[ \text{reaction distance} = 3 \cdot \frac{\text{speed}}{10} \]
\[ \text{braking distance} = \frac{\text{speed}}{10} \cdot \frac{\text{speed}}{10} \]

1.2 Task and program requirements

Implement a Java program that calculates the reaction distance, the braking distance and the overall stopping distance (in meter) for a speed entered by the user and output the results on the screen.

2 Compound interest calculation

2.1 Introduction

Calculate the value of an investment after 10 years if an amount of CHF 2000 is deposited into a savings account at an annual interest rate of 2%.

2.2 Problem definition and program requirements

Implement a Java program that calculates the amount of interest received in each year and the balance after the interest credit to the account. The output for each year shall look as follows:

In year x, you get an interest of CHF xx. New balance: CHF xxx
2.3 Extension

Enable the user to enter the parameters: investment amount and interest rate.

3 ATM (cash machine)

3.1 Introduction

The purpose of this task is to store values in variables. Furthermore, two different division operators will be used.

3.2 Problem definition

In this task, you shall simulate a ATM. The client should be able to enter how much money he wants to withdraw from his bank account. The ATM must calculate how many and which banknotes (denominations of 100, 50, 20 and 10 francs) should be output. The number of variables used should be kept as small as possible by reusing them.

The output of the amount 571 could look as follows:

<table>
<thead>
<tr>
<th>How much do you want to withdraw? 571</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entered amount of money: 571 Fr.</td>
</tr>
<tr>
<td>100-franc notes: 5</td>
</tr>
<tr>
<td>50-franc notes: 1</td>
</tr>
<tr>
<td>20-franc notes: 1</td>
</tr>
<tr>
<td>10-franc notes: 0</td>
</tr>
<tr>
<td>Remainder: 1</td>
</tr>
</tbody>
</table>

3.3 Intermediate steps

- Create a user input for an arbitrary amount of money and store the value in a variable.
- Define a variable for each kind of banknote (100, 50, 20, 10).
- Calculate how many 100-franc notes must be output, and print the value on the screen.
**Integer value of a division**

You can use \( a \div 100 \) to get the integer value of the division of \( a \) and 100. Example: \( 571 \div 100 = 5 \).

- Calculate the remaining amount.

**Integer remainder of a division**

You can use \( a \mod 100 \) to get the remainder of the division of \( a \) and 100. Example: \( 571 \mod 100 = 71 \).

- Calculate the number of all other banknotes in the same manner as for the 100-francs notes.

**Tip:** Copy the command block of 100-franc notes and modify it for the other notes.

---

### 3.4 Extension

For this extension, you need a **condition check** which will be introduced in the next module.

**Conditional program execution**

**Syntax:** The command is only executed if the condition applies.

```plaintext
if (Condition) {
    Command
}
```

- Check after the input of the amount of money whether the amount has to be rounded down and inform the client about the amount actually paid out.
- Make sure that only the banknotes which are paid out are displayed.
- Assume that only a certain maximum amount may be withdraw. For this reason, you should check whether an amount desired exceed this limit and inform the client accordingly if this is the case.
• The client may want to have banknotes of a smaller denomination. Ask the client whether he desires mixed banknotes (e.g. no = 0 and yes = 1). First, decide how you want to assemble the banknotes and then adapt the program accordingly.