Errata for

Java and Algorithmic Thinking

for the Complete Beginner

1.2 What it Hardware?

➢ **The Central Processing Unit (CPU)**
This is the part of a computer that actually performs all the tasks defined in a program **(basic arithmetic, logical, and input/output operations.)**

Exercise 13.1-2 Finding the Sum of Digits

First Approach

Step	Statement	Notes	number	digit1	digit2	digit3	digit4	r	sum
…	…		…	…	…	…	…	…	…
8	`sum = digit1+digit2+digit3+digit4`		6753	6	7	5	3	53	**21**
9	`System.out.println(sum)`	Value **21** is displayed							

Second Approach

Once more, let's try to understand the second approach using an arithmetic example. Take the same number, 6753, for example.

Fourth digit = 3	The fourth digit can be isolated if you divide the given number by 10 to get the integer remainder `digit4 = 6753 % 10`
Remaining digits = 675	The remaining digits can be isolated if you divide the given number by 10 to get the integer quotient `r = (int)(6753 / 10)`
Third digit = 5	The third digit can be isolated if you divide the remaining digits by 10 to get the integer remainder `digit3 = 675 % 10`
Remaining digits = 67	The remaining digits are now `r = (int)(675 / 10)`
Second digit = 7	The second digit can be isolated if you divide the remaining digits by 10 to get the integer remainder `digit2 = 67 % 10`
First digit = 6	The last remaining digit, which happens to be the first digit, is

	`digit1 = (int)(67 / 10)`

13.2 Review Exercises

5. Write a Java program that prompts the user to enter an integer representing an elapsed time in seconds and then displays it in the format “WW weeks DD days HH hours MM minutes and SS seconds.” For example, if the user enters the number 2000000, the message “3 weeks 2 days 3 hours 33 minutes and 20 seconds” should be displayed.

Exercise 25.1.3 Designing the Flowchart and Counting the Total Number of Iterations

Now, let’s create a trace table to observe the flow of execution.

Step	Statement	Notes	i	
1	`i = 1`		1	
2	`while (i != 6)`	This evaluates to `true`		1st Iteration
3	`i += 2`		3	
4	`while (i != 6)`	This evaluates to `true`		2nd Iteration
5	`i += 2`		5	

6	`while (i != 6)`	**This evaluates to `true`**	
7	`i += 2`		**7**
8	`while (i != 6)`	This evaluates to `true`	
9	…	…	
10	…	…	

3rd Iteration

.

.

29.8 Converting from a While-Loop to a For-loop

Exercise 29.8-3 Converting the Java Program

File_29_8_3b.java

```
public static void main(String[] args) throws java.io.IOException {
  int i;
  double s;

  s = 0;
  ~~i = 1;~~
  for (i = 1; i <= 5; i++) {
    s = s + Math.pow(i + 1, 2);
  }

  System.out.println(s);
}
```

37.4 Sorting Lists

Exercise 37.4-1 The Bubble Sort Algorithm – Sorting One-Dimensional Lists with Numeric Values

Fifth pass

1st Compare

*Elements at index positions 4 and 5 are compared. Since the value 49 is **not** less than the value **25**, **no** swapping is done.*

40.3 Formal and Actual Arguments

Remember! *There is a one-to-one match between the formal and the actual arguments. The value of argument* `a` *is passed to argument* `n1`*, the value of argument* `b` *is passed to argument* `n2`*, and so on. Moreover, the data type of the formal and the data type of the corresponding actual argument must match. You cannot, for example, pass a string to an argument of type integer!*

43.1 Simple Exercises with Subprograms

Exercise 43.1-5 How Many Times Does Each Number of the Dice Appear?

```
//Variable n1 is assigned the number of times that value 1 exists in array a
n1 = search_and_count(1, a);
//Variable n2 is assigned the number of times that value 2 exists in array a
n2 = search_and_count(2, a);
.
.
.
//Variable n6 is assigned the number of times that value 6 exists in array a
n6 = search_and_count(6, a);
```

```
//Display how many times each of the six numbers appears in array a
System.out.println(n1 + " " + n2 + " " + n3 + " " + n4 + " " + n5 + " " + n6);

//Find maximum of n1, n2,… n6
max = n1;
max_i = 1;

if (n2 > max) {
  max = n2;
  max_i = 2;
}
if (n3 > max) {
  max = n3;
  max_i = 3;
}
.
.
.
```

43.2 Exercises of a General Nature with Subprograms

Exercise 43.2-3 Progressive Rates and Electricity Consumption

File_43_2_3.java

```
…

static double find_amount(int kwh) {
  double amount;

  if (kwh <= 400) {
    amount = kwh * 0.08;
  }
  else if (kwh <= 1500) {
    amount = 400 * 0.08 + (kwh - 400) * 0.22;
  }
  else if (kwh <= 3000) {
    amount = 400 * 0.08 + 1100 * 0.22 + (kwh - 1500) * 0.35;
  }
  else {
    amount = 400 * 0.08 + 1100 * 0.22 + 1500 * 0.35 + (kwh - 3000) * 0.50;
  }

  amount += 0.26 * amount;
  return amount;
}

…
```