

CSC1016S Assignment 2

Basic Syntax - Introduction to Objects and Java

Assignment Instructions

The objective of this assignment is to introduce Java and object-oriented thinking. It involves building, testing and debugging simple programs that create and manipulate compositions of objects, and in the process, mapping your Python knowledge to Java.

The exercises are relatively simple and could easily appear in some form as Python problems in CSC1015F. An important distinction is that, moving to object-oriented programming, we aim for a richer representation of the concepts at play. We have classes of object for the things we want to 'talk' about, rather than using Strings or integers or real numbers.

Exercise One [50 Marks]

This exercise involves the use of Time and Duration classes of object that may be described as follows:

Class Time

A Time object represents a twenty-four-hour clock reading composed of hours, minutes and seconds.

Constructors

Time(String reading)

*// Create a Time object from a string representation of a twenty-four-hour clock reading
// of the form 'hh:mm[:ss]' e.g. "03:25", "17:55:05".*

Methods

public Duration subtract(Time other)

// Set the first, middle and last names of this Student object.

public String toString()

// Obtain a String representation of this Time object in the form "HH:MM:SS".

Class Duration

A Duration object represents a length of time (with millisecond accuracy).

Methods

public long intValue(String timeunit)

*// Obtain an integer value that represents as much of this duration object that can be expressed
// as a multiple of the given time unit
// For example, given a duration object d that represents 1 hour, 4 minutes, and 30 seconds,
// d.intValue("minute") produces 64.
// Permissible units are "millisecond", "second", "minute", "hour" and "day".*

This style of describing types of object will often be used in the course. A specification gives:

- The name of the class (type) of object and a brief description of what the objects represent.
- A description of constructors, the ways of creating instances of the type of object.
- A description of methods, the ways of manipulating instances of the type of object.

The description of a constructor or method comprises the signature (name, formal parameters), and the behaviour i.e. what it does.

For the Time class we've listed one way of creating it: a Time object can be created from a string representing a particular 24 hour clock time. The following code snippet provides an example:

```
Time t;  
t = new Time("13:45");
```

The code consists of two statements. The first declares a variable called 't' that will refer to a Time object. The second uses an object creation expression, and assigns the result to the variable.

(Variable declaration and assignment can actually all be done in one statement, and that is what you'll typically see in code samples in future.)

We've given two methods of manipulating a Time object, one is called 'subtract' and the other 'toString'. The subtraction method provides a means for subtracting one time from another to obtain a duration i.e. obtaining the period between them. The other method provides a means of obtaining a string representation of the object i.e. something printable.

Say we had two Time objects referred to as 't1' and 't2', we could subtract t2 from t1 using the expression 't1.subtract(t2)'

The subtract method returns a value that is a Duration object.

We've given you one method for Duration objects, 'intValue'. The method accepts a String parameter which must be the name of a time unit. It returns an integer representing the number of that unit closest to the duration value. An example is given in the specification: given a duration object, d, that represents 1 hour, 4 minutes, and 30 seconds, the expression d.intValue("minute") evaluates to 64.

Now for your task: On the Amathuba page for the assignment you will find Time and Duration classes. Download these and use them to write a Java program called CalculateDuration.java that accepts as input two 24-hour clock times, that calculates the period between them, and then prints the result as a quantity of minutes.

You may assume the second time entered always occurs at or after the first.

Sample IO (The input from the user is shown in **bold font** – do not program this):

Enter time A:

3:09

Enter time B:

16:59

The time 16:59:00 occurs 830 minutes after the time 03:09:00.

Your program must make use of all the constructors and methods specified.

Exercise Two [50 Marks]

This exercise involves the use of Money and Currency objects.

Class Money

An object of this class represents an amount of money in a particular currency. Amounts can be added and subtracted. The amount is stored as a quantity of the minor unit of the currency e.g. 1 Rand will be stored as 100 cents.

Constructors

```
public Money(String amount, Currency currency)
    // Create a Money object that represents the given amount of the given currency.
    // The String is assumed to have the following format: <currency symbol><quantity of
    // units>.<quantity of minor units> e.g. in the case of USD, $50.30, $0.34.
```

Methods

```
public Money add(Money other)
    // Add the other amount of money to this amount and return the result. The objects must be of the
    // same currency.

public String toString()
    // Obtain a string representation of the monetary value represented by this object e.g. "€45.10" for
    // a Money object that represents 45 euros and 10 cents.
```

Class Currency

An object of this class represents a Currency such as US Dollars or British Pound Stirling.

A currency has an ISO 4217 currency code and a symbol denoting the currency's major unit. Many currencies have a minor (or fractional) unit such as the cent in the case of US dollars.

A currency object provides facilities for creating strings and for interpreting strings that represent amounts of the currency.

It is assumed that the currency symbol always appears in front of an amount; that negative amounts are represented by a minus sign, '-', that precedes the currency symbol, "-£34.50" for example; that the decimal point is always represented using a full stop; that no attempt is made to group the digits of large quantities, so for example, one million Rand is assumed to be represented as "R1000000" (as opposed to "R1,000,000").

Constructors

```
public Currency(String symbol, String code, int minorPerMajor)
    // Create a Currency object that represents the currency with the given unit symbol (e.g. "£" for
    // Sterling), ISO 4217 code, and number of minor units per major units (e.g. 100 in the case of
    // pennies per British Pound).
```

Consider the following code snippet:

```
Currency rand = new Currency("R", "ZAR", 100);
Money money = new Money("R14.50", rand);
System.out.println(money.toString());
```

The code creates a Currency object that represents ZAR (South African Rand), and a Money object that represents the amount R14.50. It uses 'toString()' to get a string representation of the Money object that it then prints out.

Now for your task: On the Amathuba page for the assignment you will find Money and Currency classes. Download these and use them to write a Java program called "SumCosts.java" that asks the user to enter a series of Rand amounts, sums them, and prints the result.

Sample IO (The input from the user is shown in **bold font** – do not program this):

```
Enter an amount or '[D]one' to quit:
```

```
D
```

```
Total: R0.00
```

Sample IO (The input from the user is shown in **bold font** – do not program this):

```
Enter an amount or '[D]one' to quit:
```

```
R0.99
```

```
Enter an amount or '[D]one' to print the sum and quit:
```

```
R15
```

```
Enter an amount or '[D]one' to print the sum and quit:
```

```
R17.95
```

```
Enter an amount or '[D]one' to print the sum and quit:
```

```
D
```

```
Total: R33.94
```

You must use the classes and methods described.

Your program will require a 'while-loop'. The Java syntax is very similar to that of Python. Here's an example:

```
int i=0;
while (i<10) {
    i=i+1;
    System.out.println(i);
}
```

The code prints out, one per line, the values 1 to 10 .

Unlike Python, the terminating condition is enclosed in brackets. The body of the while loop – the code to be executed – is enclosed in curly braces.

NOTES/HINTS:

- To determine whether one string, s1, is equal to another, s2, use the 'equals' method e.g. 's1.equals(s2)'.
- The string method 'charAt' accepts an index value as a parameter and, when applied to a string, returns the character at that position e.g. the expression "CSC1016S".charAt(3) produces '1'.
- The expression "CSC1016S".charAt(2)=='f' is false while the expression "CSC1016S".charAt(1)=='S' is true.

Submission

Submit the ZIP file *bearing your student number* containing `CalculateDuration.java` and `SumCosts.java` programs to the automatic marker.

The automatic marker will use custom versions of the `Time`, `Duration`, `Money` and `Currency` classes to determine whether you have used them as intended.