

Take Assessment: Practical Quiz 1

This is a resumption of your previous attempt.

Please note: At this time, we cannot track responses you entered previously.

Please answer the following question(s).

If the assessment includes multiple-choice questions, click the "Submit Answers" button when you have completed those questions.

Please complete this assessment by Mon Nov 12 2007 23:06:30 GMT+0800.

1. [Go to bottom of question.](#)

iCarnegie Information

Prerequisites, Goals, and Outcomes

Prerequisites: Before you begin this exercise, you need mastery of the following:

- *Java Language*
 - Use of Java control structures
- *Java API*
 - Knowledge of package java.io
 - How to get input from the keyboard
 - How to output results, prompts, and errors
 - Knowledge of wrapper class Integer: how to read an integer value as input
- *Exception Objects*
 - Knowledge of exception handling
 - Knowledge of NumberFormatException
- *Programming Practice*
 - How to produce Java applications
 - Knowledge of Sun's code conventions
 - Knowledge of Javadoc: how to document classes, methods, and variables

Goals: Reinforce your ability to build console applications using I/O classes, wrapper classes, and exceptions

Outcomes: You will master the following skills:

- Produce a menu-driven application that reads data from the keyboard and displays results in the standard output
- Produce an application that uses:
 - Wrapper classes to read numerical data
 - Exceptions to handle malformed data
 - Control structures to control the reading of data

Background

Menus for soliciting input from the user are common in console applications. This assignment asks you to construct a menu-driven application that reads and validates user input. Your application will consist of one class called `ICarnegieInfoApplication`.

Description

`ICarnegieInfoApplication` presents the user with a menu of options, prompts the user for a choice, and processes the user's response:

- Choice 0 terminates the program.
- Choice 1 displays the name of *iCarnegie*.
- Choice 2 displays the address of *iCarnegie*.
- Choice 3 displays the telephone of *iCarnegie*.
- Choice 4 displays the email of *iCarnegie*.
- Choice 5 displays the URL of *iCarnegie*.

The class `ICarnegieInfoApplication` uses the class `ICarnegieInfo`. `ICarnegieInfo` stores the contact information for *iCarnegie*. A complete implementation of `ICarnegieInfo` is provided in the student archive [student-files.zip](#). Review its documentation and become familiar with it.

- [ICarnegieInfo](#). Documentation for class `ICarnegieInfo`

Note that class `ICarnegieInfo` does not have a public constructor. Instead, you will use the method `ICarnegieInfo.getInstance` to obtain an instance of the class.

A partial implementation of class `ICarnegieInfoApplication` is provided. This class contains the following methods:

- *public static void main(String[] args)*. This method is complete and requires no modification. `main` has a loop that calls the method `getChoice` to obtain the user's choice. `main` then displays the information requested. Choice 0 causes the loop to terminate gracefully.
- *private static int getChoice()*. This method must be completed. `getChoice` presents the user with a menu of options and verifies the user's choice.

Files

The following files are needed to complete this assignment:

- [exe-icarnegie-info.jar](#). Download this file. It is a sample executable.
- [student-files.zip](#). Download this file. This archive contains the following:
 - *ICarnegieInfo.class*. A complete implementation
 - *ICarnegieInfo.html*. Documentation
 - *ICarnegieInfoApplication.java*. Use this template to complete your implementation.

Tasks

To complete this assessment, you will implement method `getChoice`. The following steps will guide you through this assignment. Document using Javadoc and follow Sun's code conventions. Work incrementally and test each increment. Save often.

1. **Extract** the student archive by issuing the following command at the command prompt:

```
C:\>unzip student-files.zip
```

2. **Run** the sample executable by issuing the following command at the command prompt:

```
C:\>java -jar exe-icarnegie-info.jar
```

Observe how the program responds to the following types of input:

- A choice that is not an integer: `aa`

- An integer that is below the valid range: -1
- An integer that is above the valid range: 6
- An integer that is within the valid range: 1, 2, 3, 4, 5 and 0

3. **Next**, complete the implementation of method `getChoice`:

protected static int getChoice(). This method displays the following menu of options and prompts the user for a choice.

```
0 - Quit
1 - Display name
2 - Display address
3 - Display telephone
4 - Display email
5 - Display URL
choice>
```

`getChoice` then reads and validates the user's input:

- If the user enters a choice that is not an integer, [java.lang.NumberFormatException](#) is caught and output.
- If the user enters an integer outside of the valid range [0,5], an error message is displayed.

If the input is invalid, `getChoice` re-prompts the user for a new choice. If the input is valid, `getChoice` returns the user's choice.

Submission

Upon completion, submit **only** the following:

1. *ICarnegieInfoApplication.java*

[Go to top of question.](#)

File to submit:

Upload File

Forward File

Refresh

Ready for Grading

2. [Go to bottom of question.](#)

SecondsCalculator

Prerequisites, Goals, and Outcomes

Prerequisites: Before you begin this exercise, you need mastery of the following:

- *Java Language*
 - Knowledge of Java control structures
- *Java API*
 - Knowledge of package `java.io`
 - How to get input from the keyboard
 - How to output results, prompts, and errors
 - Knowledge of wrapper class `Integer`: how to read an integer value as input
 - Knowledge of class `StringTokenizer`: how to parse a line with multiple values
- *Exception Objects*
 - Knowledge of exception handling
 - Knowledge of `NumberFormatException`
- *Programming Practice*
 - How to produce Java applications
 - Knowledge of Sun's code conventions
 - Knowledge of Javadoc: how to document classes, methods, and variables

Goals: Reinforce your ability to use I/O classes, tokenizers, wrapper classes, and exceptions

Outcomes: You will master the following skills:

- Produce an application that reads input from the keyboard and uses:
 - Wrapper classes to read numerical data
 - `StringTokenizer` to parse a line with multiple values
 - Exceptions to handle malformed data
 - Control structures to control the reading of data

Background

This assignment asks you to construct a simple application that processes a line of text that contains multiple values. Your application will read a time interval expressed in hours, minutes, and seconds from the keyboard and then display the total number of seconds in the specified time interval. Your application will consist of one class called SecondsCalculator.

Description

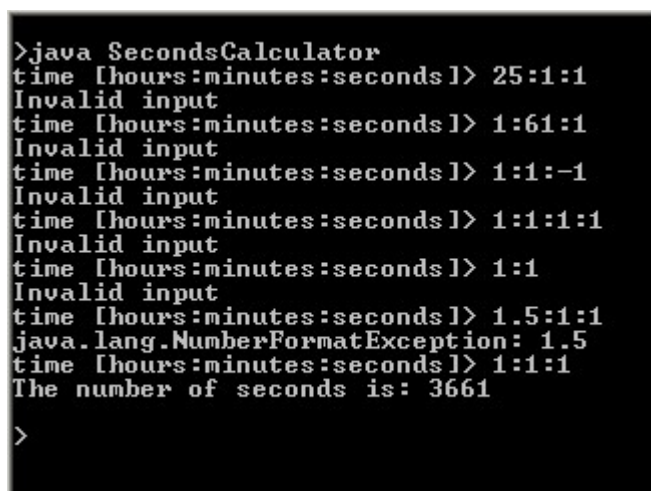
SecondsCalculator reads, from the keyboard, three integers entered on the same line and separated by a colon (:). The first integer represents the hours, the second represents the minutes, and the third represents the seconds.

If the user enters a non-integer, or an integer outside of the valid range, the application displays an error message. Either way, the user is re-prompted for new input.

If the input is valid, the application displays the total number of seconds the specified time interval. The total number of seconds for time interval *hours:minutes:seconds* is computed using the following expression:

$$\text{hours} * 3600 + \text{minutes} * 60 + \text{seconds}$$

The following is a screen shot of the application.



```
>java SecondsCalculator
time [hours:minutes:seconds]> 25:1:1
Invalid input
time [hours:minutes:seconds]> 1:61:1
Invalid input
time [hours:minutes:seconds]> 1:1:-1
Invalid input
time [hours:minutes:seconds]> 1:1:1:1
Invalid input
time [hours:minutes:seconds]> 1:1
Invalid input
time [hours:minutes:seconds]> 1.5:1:1
java.lang.NumberFormatException: 1.5
time [hours:minutes:seconds]> 1:1:1
The number of seconds is: 3661
>
```

Figure 1 *Execution of SecondsCalculator*

Files

The following files are needed to complete this assignment:

- [*exe-seconds-calculator.jar*](#). Download this file. It is a sample executable.

Tasks

To complete this assignment, you will implement class `SecondsCalculator`. The following steps will guide you through this assignment. Document using Javadoc and follow Sun's code conventions. Work incrementally and test each increment. Save often.

1. **Run** the sample executable by issuing the following command at the command prompt:

```
C:\>java -jar exe-seconds-calculator.jar
```

Observe how the application responds to the following types of input:

- Input with a value that is not an integer: `1.0:1:1`, `1:1.0:1`, or `1:1:1.0`.
 - Input with an integer outside of a valid range: `24:1:1`, `1:60:1`, `1:1:60`, `-1:1`, `1:-1:1`, or `1:1:-1`.
 - Input that contains more than three integers: `1:1:1:1`.
 - Input that contains fewer than three integers: `1:1`.
 - Valid input: `0:0:0`, `1:1:1`, or `23:59:59`.
2. **Then**, write this application from scratch: The application reads, from the keyboard, three integers entered on the same line and separated by a colon (:). Use [`java.util.StringTokenizer`](#) to extract the hours, minutes, and seconds from the input. The first integer represents the hours and should be confined to the range `[0, 23]`; the second integer represents the minutes and should be confined to the range `[0, 59]`; and the third integer represents the seconds and should be confined to the range `[0, 59]`.

The application validates the user's input:

- If the user enters fewer than three values, an error message is displayed.
- If the user enters more than three values, an error message is displayed.
- If the user enters a value that is not an integer,

- [java.lang.NumberFormatException](#) is caught and displayed.
- o If the user enters an integer outside the valid range, an error message is displayed.

The error messages displayed by your implementation should match the error messages displayed by the sample executable.

If the input is invalid, the application re-prompts the user for new input. If the input is valid, the application displays the total number of seconds in the specified time interval.

Submission

Upon completion, submit **only** the following:

1. *SecondsCalculator.java*

[Go to top of question.](#)

File to submit:

[Upload File](#)

[Forward File](#)

[Refresh](#)

[Ready for Grading](#)

[Go to top of assessment.](#)

© Copyright 2006 iCarnegie, Inc. All rights reserved.