12/3/2015

**Agile Board Application Report**

Candidate: Jun Feng  
Degree: BSc in I.T.  
University of Technology, Sydney

Table of Contents

1 Overview 3

2 Assumption 3

3 Application Design 3

3.1 System Structure 3

3.1.1 Model Package, scrumBoardComponents 3

3.1.2 Dirver Package, scrumBoardManagement 4

3.2 User Friendliness 4

3.3 System Robustness 4

3.3.1 IDGenerator and IDType Classes 4

3.3.2 TaskStatus Classes 4

3.3.3 Input and Exception Handling 4

4 Application Functions and Screenshots 5

4.1 Create a story 5

4.2 List stories 5

4.3 Delete a story 6

4.4 Complete a story 6

4.5 Create a task 7

4.6 List tasks 7

4.7 Delete a task 8

4.8 Move a task 8

4.9 Update a task 9

4.10 Exit 9

5 Conclusion 9

Appendix A: How to run the application in the Terminal 10

# Overview

This is a console application simulating the agile scrum task board. Users can create stories, tasks and move them as their work progresses. It features using menu choice to reduce command input, providing timely system feedback via messages and possessing well-designed exception handling against users’ invalid input.

This application is implemented in Java and compiled with JDK 1.6. It is packaged as a zip file, and instruction of how to run it in the terminal is in the Appendix.

In order to version control and maintain the source code, a Github repository has been set up at <https://github.com/Elfpope/IR_Agile_Scrum_Board>

# Assumption

The deleted story would be wiped out from the system, hence can no longer be listed or processed by the application. And the same rule applies to the deleted task.

The completed story could not be added in more tasks and its tasks could not be processed any more. But it can be listed and deleted.

# Application Design

A good design of the application can enhance system structure, user friendliness and system robustness. All these three factors have been taken into account along with the development.

## System Structure

The structure of this application has been divided into two packages. One is the model package, “scrumBoardComponents”; the other is the driver package responsible for system control and user interaction, “scrumBoardManagement”.

### Model Package, scrumBoardComponents

The model package uses object-orientated programming (OOP) to map three main classes in the system, including Board, Story and Task. They have been well defined by having ArrayList to store data and encapsulating methods at the correct level to match up the business logic of the agile scrum board. Apart from these, another three classes serve the purpose of system robustness, including IDGenerator, IDType, and TaskStatus, which are to be soon explained in the later section.

### Dirver Package, scrumBoardManagement

The driver package contains the application components of control functions and user interactions. The class of ScrumBoardApplication(SBA) is created solely for the system control. Even though the control logic in SBA could have been pushed into Board class as there is only one board, it would not be the best practice of OOP. Furthermore, in terms of future scalability, if users need for a few new boards to manage different teams or projects later, an ArrayList of boards could be easily added plus list management methods without making any changes to the Board class in the model package. Also, methods of reading user input and validating the input are put together into one class, “InputScanner”.

## User Friendliness

In order to facilitate the command input for operating the system, user can select the desired function by entering only one character from the printed out menu, which can also greatly reduce typos from user.

Each time when user selects a choice, either successful or not, the result will print out as the system feedback. So user could ensure the expected operation has completed or needs to review the input and redo.

## System Robustness

Since the application is based on command line input, user typo could be fatal to system failure. There are three mechanisms implemented to improve system robustness, which are IDGenerator/IDType class, TaskStatus Classes and Input/exception handling.

### IDGenerator and IDType Classes

IDGenerator class is designed to automatically generate an ID for either a new story or a new task. To distinguish the two types of ID, an alphabet letter ‘S’ or ‘T’ will also attached as an initial to generated ID. This way of ID generation could maintain the format of the ID, ensure its consistency and beneficiate story/task manipulation in the other operation.

On the other hand, IDType, an enum class, is used to reduce code repeat. As some similar operations amongst story and task, it could consolidate the similar operations into one method by distinguishing them by IDType.

### TaskStatus Classes

TaskStatus is an enum classes. One character choice to select the desired task status has also been implemented to avoid inconsistent status input and reduce user typo. Additionally, it has a task transition guard implemented to ensure status changes correctly.

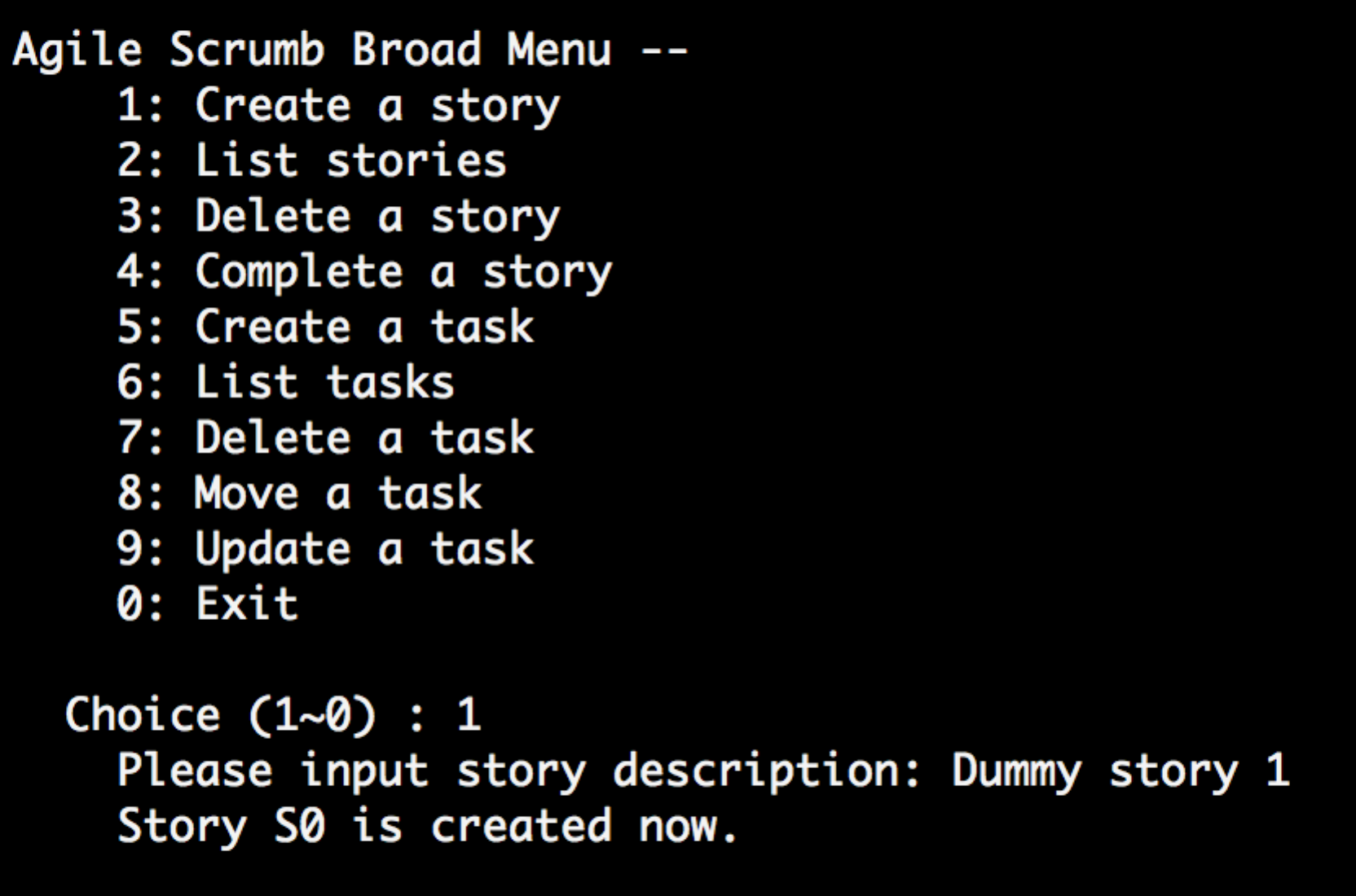
### Input and Exception Handling

Along the development of this system, input and exception handling has been considered at all time, mentioned as below:

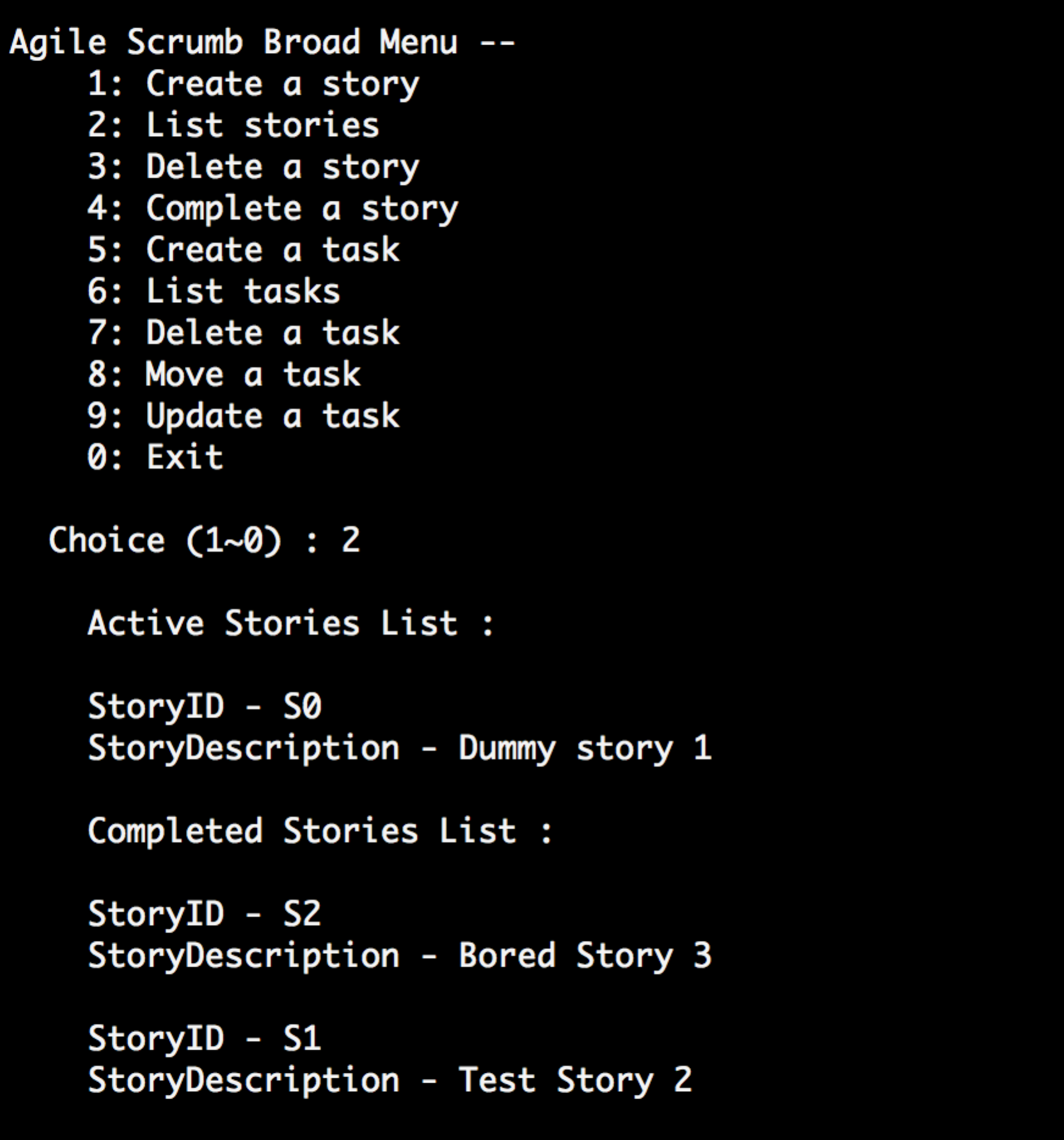
* One character input of menu choice to operate the system
* ID automate generator forces user to follow the pre-defined ID format
* One character input of task status selection to move the task
* Warning/error message will be printed out when corresponding invalid inputs or exceptions are caught.

# Application Functions and Screenshots

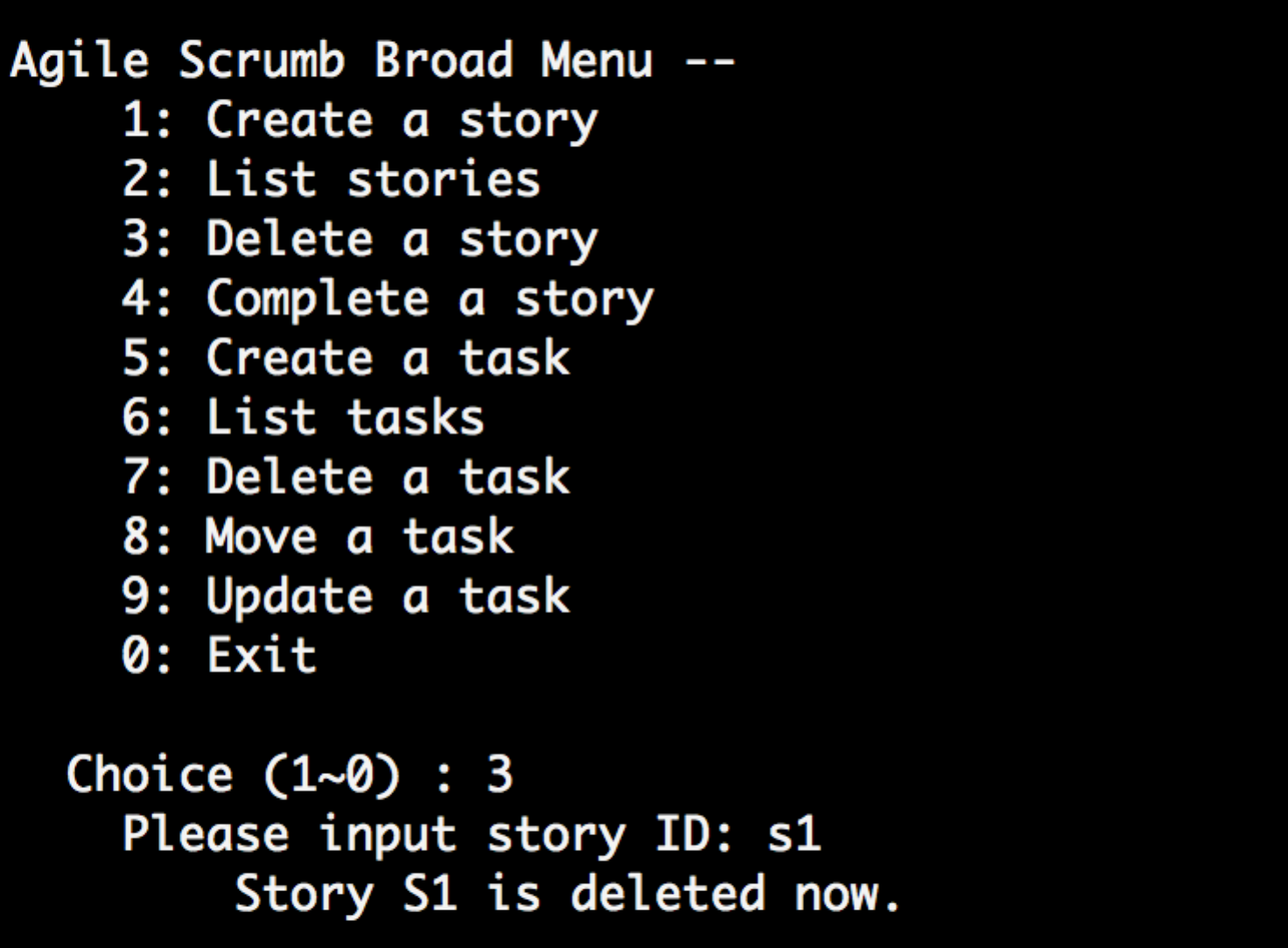
## Create a story



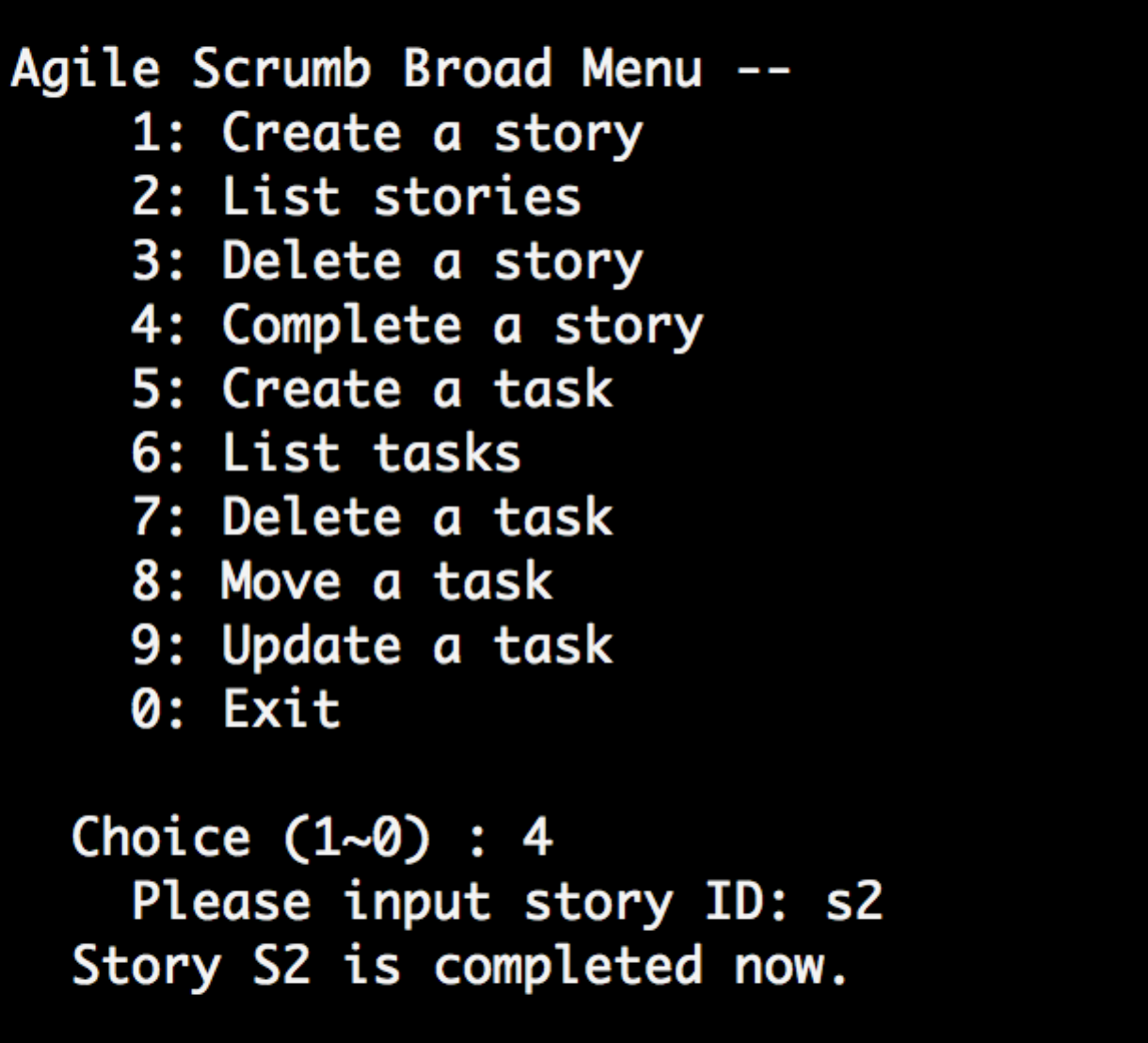
## List stories



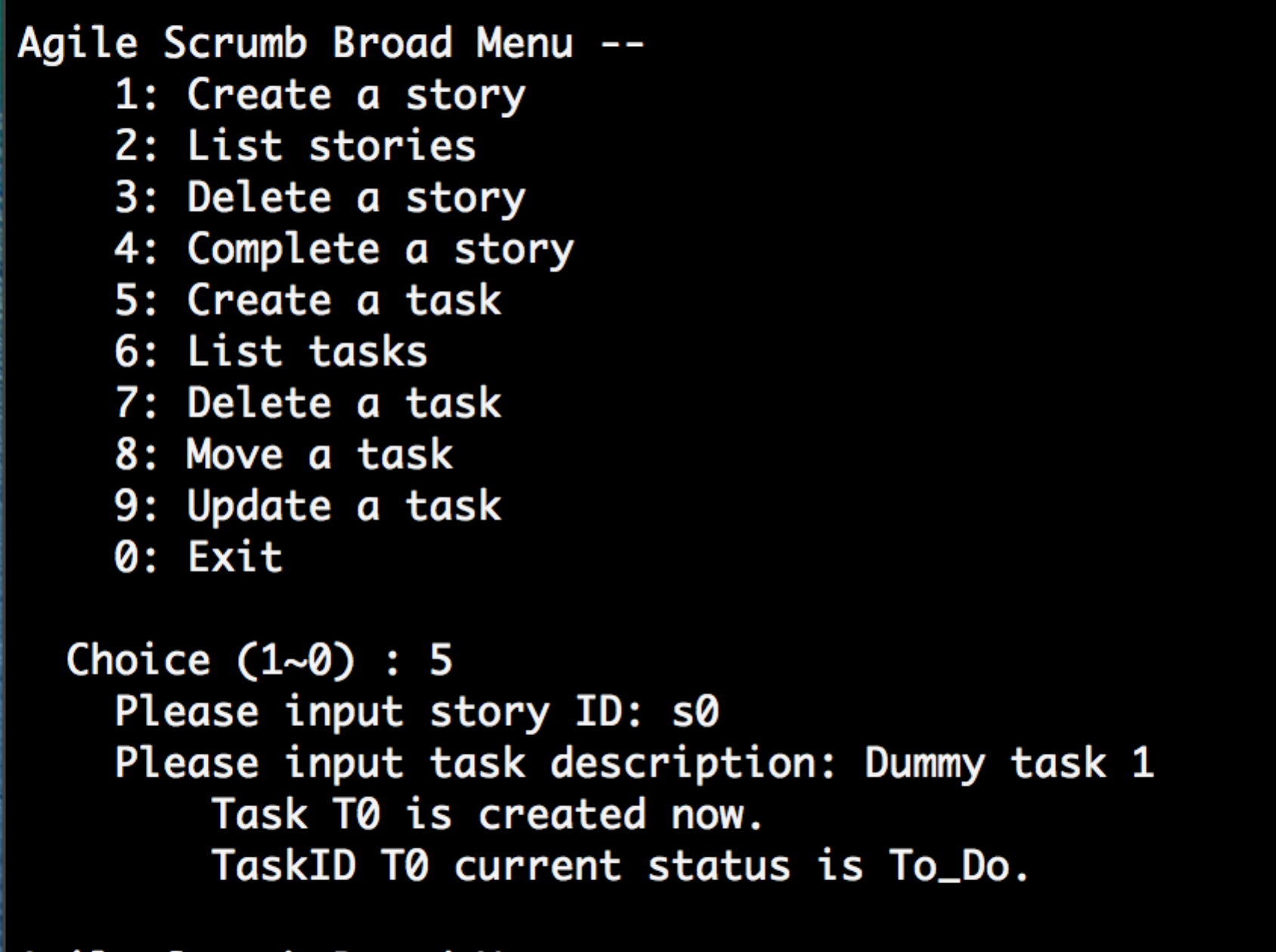
## Delete a story



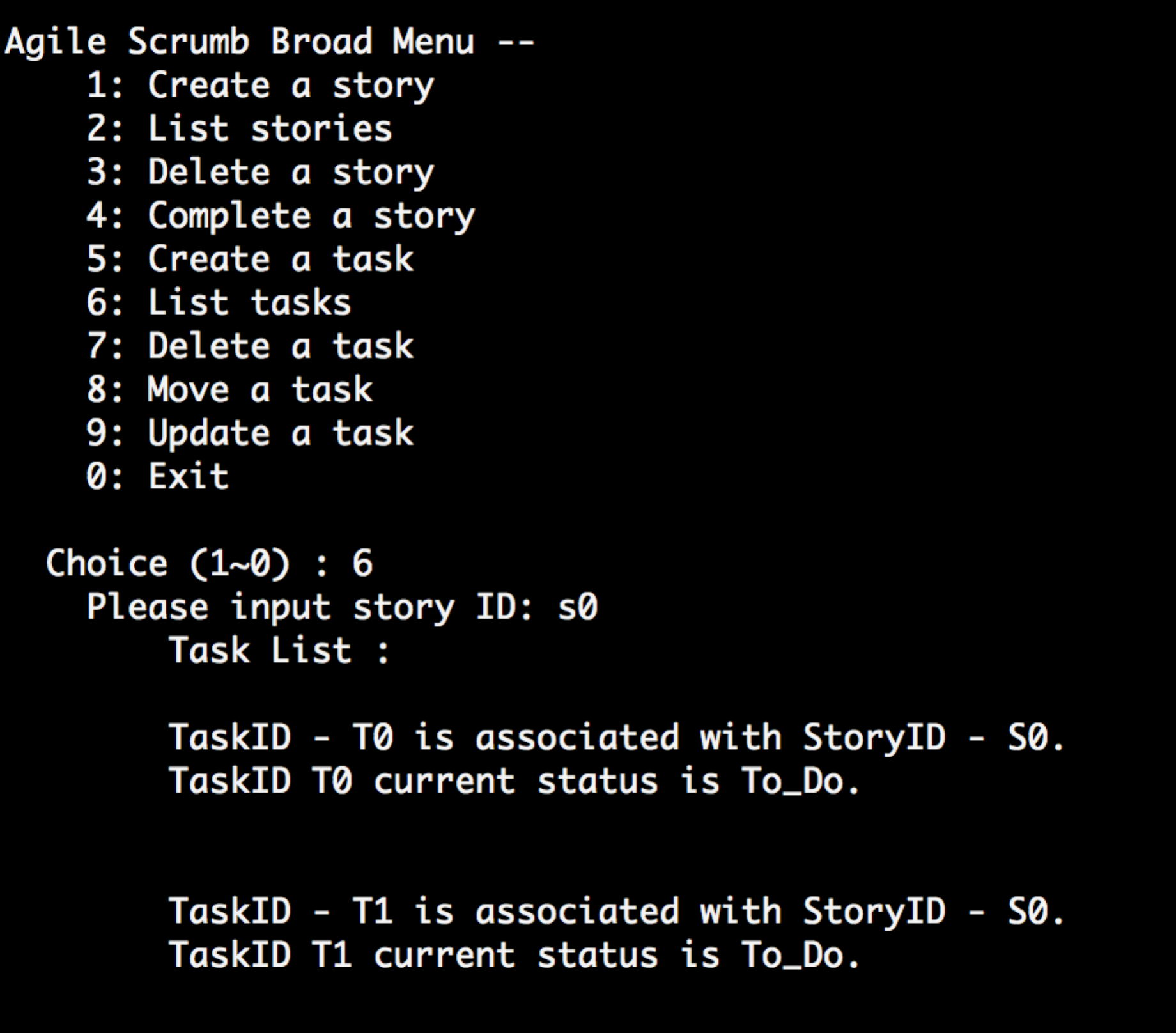
## Complete a story



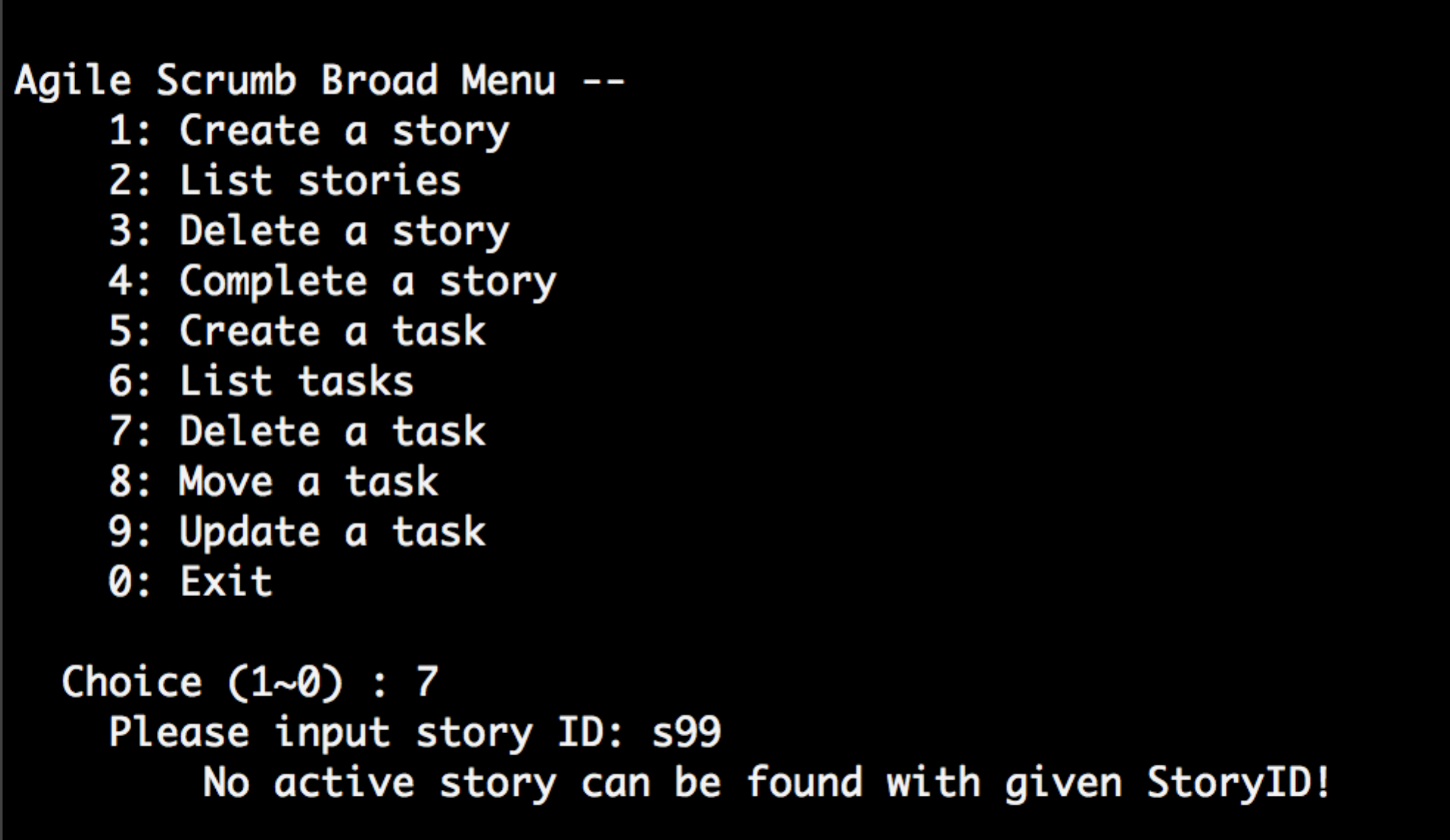
## Create a task



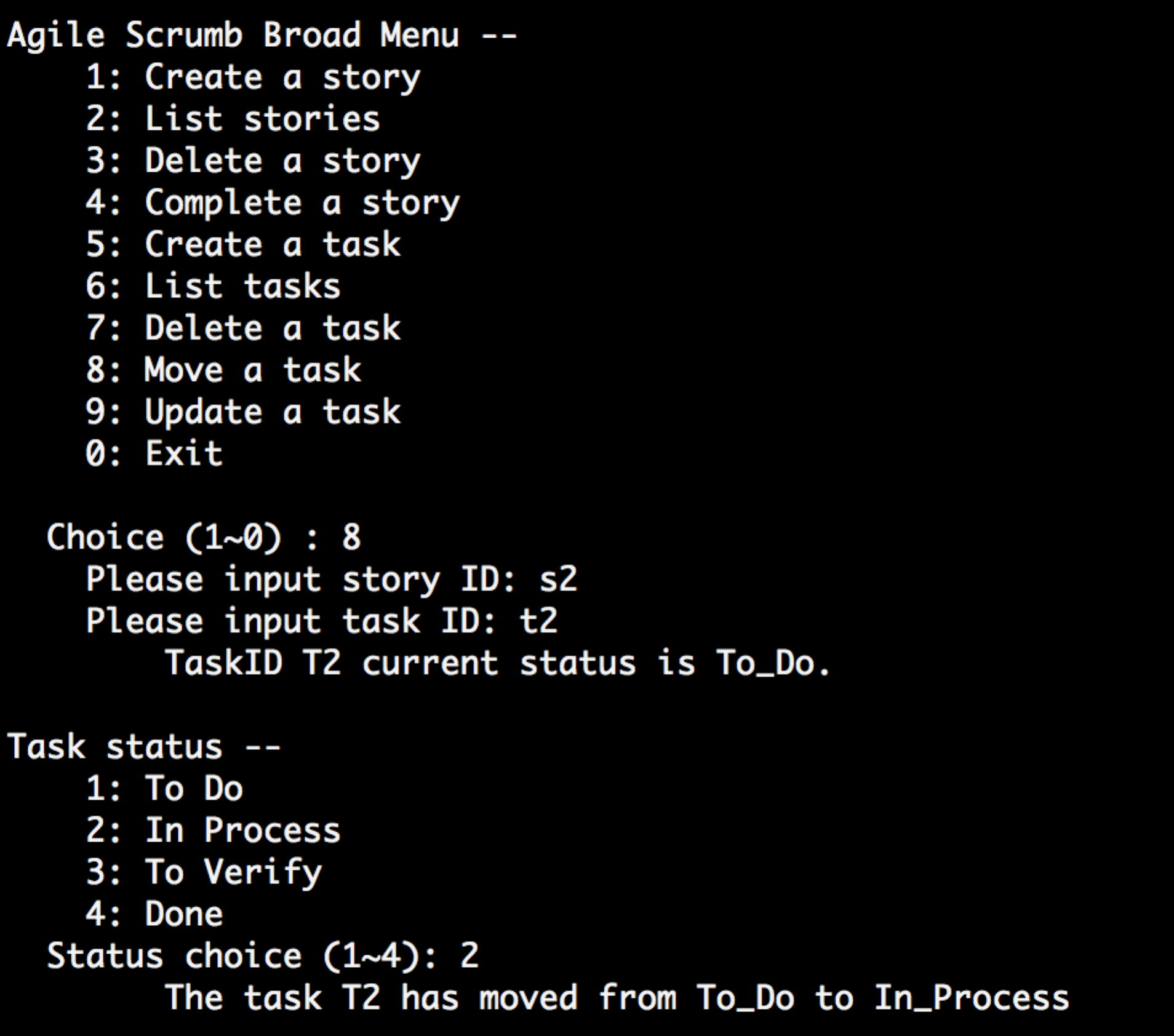
## List tasks



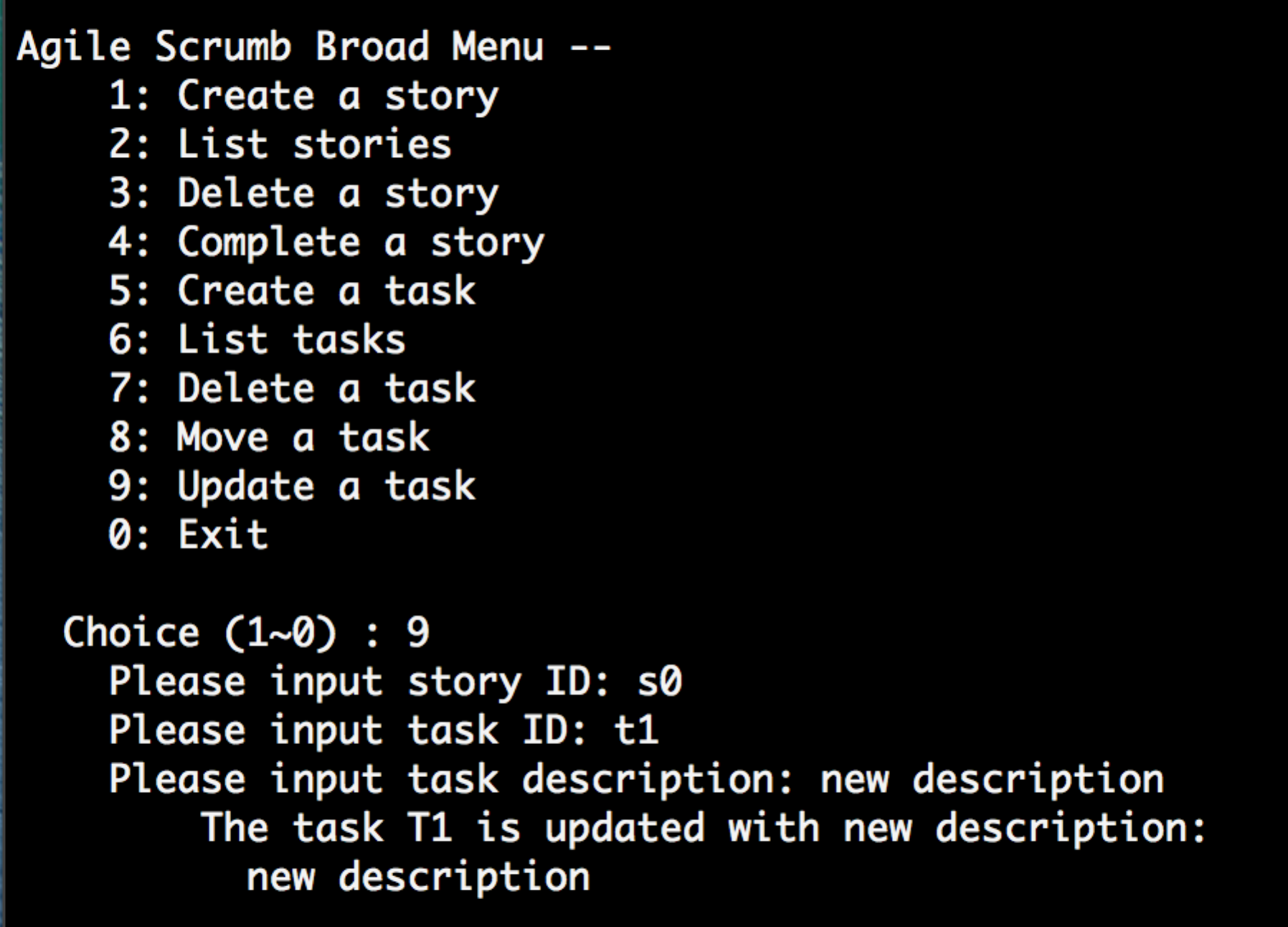
## Delete a task



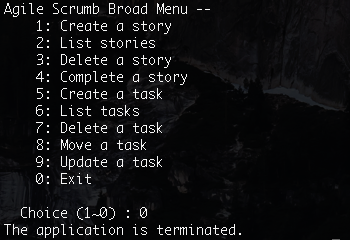
## Move a task



## Update a task



## Exit



# Conclusion

This application has been developed to match up the user requirement. Effective communication between the client (IR) and the developer has been established via email to help clear up some ambiguousness in the requirement. Besides, sufficient tests have carried out to ensure the system functions as expected. Further improvement could be followed depending on the client or user’s feedback.

# Appendix A: How to run the application in the Terminal

a. Navigation

locate the IR\_Board.zip in the terminal

b. Unzip under the current directory with below command

unzip IR\_Board.zip -d ./IR\_Board

c. Create a bin directory with below command

mkdir ./IR\_Board/bin

d. Compile (in IR\_Board directory) with below command

javac -d bin -sourcepath src src/scrumBoardManagement/ScrumBoardApplication.java

e. Execute with below command

java -cp bin scrumBoardManagement.ScrumBoardApplication