Requirements Specification

1. Scope

1.1 Overview

The Command Line Utility component simplifies the process of providing and parsing switches for Java command line applications. A switch is defined as an additional argument to a program or script that provides additional processing rules at runtime to give the user more flexibility and control. This component defines interfaces to easily define the list of allowable switches, to parse the switches at runtime and to validate additional arguments.

1.2 Logic Requirements

1.2.1 Definition

- Define a method by which a class can register a list of command line switches.
- Switches can be defined with 0, 1 or many required arguments.
- Define a method by which to obtain the list of available switches.

1.2.2 Parsing and Validation

- Accept the array of Strings representing the switches.
- Parse the array and return the details to the calling client.
- Enable validation rules to be configured and executed for each switch.

1.3 Example of the Software Usage

An example of where switches are used is in the java executable itself. Typing "java -help" on the command line will yield the list of accepted switches. These switches allow configuration of runtime properties, for example, configuration of the CLASSPATH.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

None specified.

2.1.3 Environment Requirements

- Development language: Java1.4
- Compile target: Java1.3, Java1.4

2.1.4 Package Structure

Com.topcoder.util.commandline

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

None.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

None.

3.2.2 Third Party Component, Library, or Product Dependencies:

None Specified.
**Please review the TopCoder Software component catalog for existing components that can be used in the design.

3.2.3 QA Environment:

- Solaris 7
- RedHat Linux 7.1
- Windows 2000

3.3 Design Constraints

3.3.1 Development Standards:

- It is required that all code be clearly commented.
- All code must adhere to javadoc standards, and, at minimum, include the @author, @param, @return, @throws and @version tags.
    - When populating the author tag, use your TopCoder member handle. In addition, please do not put in your email addresses.
    - Copyright tag: Copyright © 2002, TopCoder, Inc. All rights reserved
- For standardization purposes, code must use a 4-space (not tab) indentation.
- Do not use the ConfigurationManager inside of EJB's. The usage of the java.io.* package to read/write configuration files can potentially conflict with a restrictive security scheme inside the EJB container.
- All code must adhere to the Code Conventions outlined in the following: http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html

3.3.2 Database Standards:

- Table and column names should be as follows: table_name, not tableName.
- Every table must have a Primary Key.
- Always use Long (or database equivalent) for the Primary Key.
  For maximum portability, database objects (table names, indexes, etc) should be kept to 18 characters or less. Informix and DB2 have an 18-character limit.

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification
- Test Cases

3.4.2 Help / User Documentation

- Javadocs must provide sufficient information regarding component design and usage.