Informal Specification of the Shakespeare Programming Language

1 Introduction

“Late at night sometime in February, Kalle Hasselström and Jon Åslund (that is us, we, the authors) were sitting with a programming assignment due for demonstration at nine the following morning. It was assignment number four in our Syntax Analysis course and we were pretty tired with it. The last assignment, on the other hand, seemed like much more fun, because you were allowed to do pretty much whatever you wanted as long as it involved lexical and syntactical analysis. So instead of finishing the fourth assignment, we started making up some great ideas for the fifth, the kind you only conceive of when you really should be asleep.”

2 Program

All SPL programs must begin with a title/description, followed by the character declarations, and finally the program parts.

Syntax

```
Program     ::=     Comment .
                    Declaration+
                    Part+
```

Semantics

- Programs must include a comment first, 1 or more character declarations, and 1 or more parts (also referred to as Acts).
- Comments have no value to the program except to provide some initial information about the code for readers.
- Character declarations are required, and only happen once, before the program starts.
- Parts identify the main components, or Acts, of the program, and its organization.

Example

```
The Infamous Hello World Program.

Romeo, a young man with a remarkable patience.
Juliet, a likewise young woman of remarkable grace.
Ophelia, a remarkable woman much in dispute with Hamlet.
Hamlet, the flatterer of Andersen Insulting A/S.

      Act I: Hamlet's insults and flattery.

      Scene I: The insulting of Romeo.

[Enter Hamlet and Romeo]

Hamlet:
  You lying stupid fatherless big smelly half-witted coward!
  You are as stupid as the difference between a handsome rich brave
  hero and thyself! Speak your mind!

  You are as brave as the sum of your fat little stuffed misused dusty
  old rotten codpiece and a beautiful fair warm peaceful sunny summer's
  day. You are as healthy as the difference between the sum of the
  sweetest reddest rose and my father and yourself! Speak your mind!
```

(Continued… full 'Hello World' Program in Appendix)

3 Declarations

Variables in SPL are known as characters. All characters needed by the program must be declared after the initial comment and before the program parts. Characters are initially set to a null value, and can store either single signed integers or a stack of signed integers.

Syntax

Declaration ::= Character , Comment **.**

Character ::= **Achilles** | **Adonis** | **Adriana** | **Aegeon** | **Aemilia** | **Agamemnon** | **Agrippa** | **Ajax** | **Alonso** | **Andromache** | **Angelo** | **Antiochus** | **Antonio** | **Arthur** | **Autolycus** | **Balthazar** | **Banquo** | **Beatrice** | **Benedick** | **Benvolio** | **Bianca** | **Brabantio** | **Brutus** | **Capulet** | **Cassandra** | **Cassius** | **Christopher Sly** | **Cicero** | **Claudio** | **Claudius** | **Cleopatra** | **Cordelia** | **Cornelius** | **Cressida** | **Cymberline** | **Demetrius** | **Desdemona** | **Dionyza** | **Doctor Caius** | **Dogberry** | **Don John** | **Don Pedro** | **Donalbain** | **Dorcas** | **Duncan** | **Egeus** | **Emilia** | **Escalus** | **Falstaff** | **Fenton** | **Ferdinand** | **Ford** | **Fortinbras** | **Francisca** | **Friar John** | **Friar Laurence** | **Gertrude** | **Goneril** | **Hamlet** | **Hecate** | **Hector** | **Helen** | **Helena** | **Hermia** | **Hermonie** | **Hippolyta** | **Horatio** | **Imogen** | **Isabella** | **John of Gaunt** | **John of Lancaster** | **Julia** | **Juliet** | **Julius Caesar** | **King Henry** | **King John** | **King Lear** | **King Richard** | **Lady Capulet** | **Lady Macbeth** | **Lady Macduff** | **Lady Montague** | **Lennox** | **Leonato** | **Luciana** | **Lucio** | **Lychorida** | **Lysander** | **Macbeth** | **Macduff** | **Malcolm** | **Mariana** | **Mark Antony** | **Mercutio** | **Miranda** | **Mistress Ford** | **Mistress Overdone** | **Mistress Page** | **Montague** | **Mopsa** | **Oberon** | **Octavia** | **Octavius Caesar** | **Olivia** | **Ophelia** | **Orlando** | **Orsino** | **Othello** | **Page** | **Pantino** | **Paris** | **Pericles** | **Pinch** | **Polonius** | **Pompeius** | **Portia** | **Priam** | **Prince Henry** | **Prospero** | **Proteus** | **Publius** | **Puck** | **Queen Elinor** | **Regan** | **Robin** | **Romeo** | **Rosalind** | **Sebastian** | **Shallow** | **Shylock** | **Slender** | **Solinus** | **Stephano** | **Thaisa** | **The Abbot of Westminster** | **The Apothecary** | **The Archbishop of Canterbury** | **The Duke of Milan** | **The Duke of Venice** | **The Ghost** | **Theseus** | **Thurio** | **Timon** | **Titania** | **Titus** | **Troilus** | **Tybalt** | **Ulysses** | **Valentine** | **Venus** | **Vincentio** | **Viola**

Semantics

- Characters are identified using one of the traditional names in Shakespeare's works. The name identifier must be in the list of available names.
- Characters have a signed integer value, or the keyword null. Any other types will throw an error.
- Characters can also have a stack of signed integers, although not directly accessed like the character's main value. This is not required, but any value other than a signed integer, or an empty stack, will throw an error.

Examples

a) `Romeo, a young man with a remarkable patience.`

b) `Juliet, a likewise young woman of remarkable grace.`

4 Parts

SPL programs are broken into Acts and Scenes, and must have at least one Act. Each Act must have at least one Scene.

Syntax

Part	::=	**Act** Numeral **:** Comment **.** Sub-part+
Sub-part	::=	**Scene** Numeral **:** Comment **.** Stage

Semantics

- Parts are identified with the keyword “Act” followed by a roman numeral, a semicolon, and a Comment of any ASCII characters except a period. The period denoted the end of the Part declaration.
- Part numerals must be denoted in order, starting from I, and can only be declared once.
- Sub-parts are identified with the keyword “Scene”, using the same structure as Part declarations.
- Sub-part numerals must be denoted in order, starting from I, and can only be declared once per Part. Sub-part numerals must restart at I for every new Part.
- Sub-parts contain the program code that interacts with the program Stage.

Examples

a)
```
Act I: Hamlet’s insults and flattery.
```

b)
```
Scene I: The insulting of Romeo.
```

5 Stage

Every Scene has an active stage, with characters and dialogue between them.

Syntax

```
Stage          ::=     (Presence)
                       Dialogue
                       (Presence)
```

Semantics

- Only one Stage exists in a program lifetime.
- Characters can enter and exit at the beginning or end of a Scene, denoted by Presence. If they do not exit at the end of a Scene or Act, they remain on stage as the next Act or Scene starts.
- All characters must be off the Stage at the end of the program lifetime.

Example

```
[Enter Romeo and Juliet]

Juliet:
  Thou art as cunning as a dimwitted coward. Speak your mind!

...

[Exit Juliet]
```

6 Presence

The presence of a character on stage determines its ability to have lines, as well as its ability to be referenced in the lines of other characters.

Syntax

```
Presence    ::=   [ Action ]
Action      ::=   Enter Character (& Character)
             |    Exit Character
             |    Exeunt (Character & Character)
```

Semantics

- Enter must have at least one character, and can have up to two.
- Exit must have only one character.
- Exeunt can have either both present Characters declared, or none. Both are equivalent--the Stage is cleared.

Examples

a) `[Enter Romeo & Juliet]`

b) `[Enter Horatio]`

c) `[Exit Romeo]`

d) `[Exeunt Romeo and Juliet]`

e) `[Exeunt]`

7 Dialogue

Dialogue is made up of the banter between characters on stage. This is the meat of an SPL program.

Syntax

Dialogue	::=	Line*
Line	::=	Character **:** Sentence+
Sentence	::=	Assignment | Conditional | Goto | Interaction | Stack-Op

Semantics

- Dialogue consists of Lines spoken by Characters that are present on stage. Characters must be on stage to have lines.
- Lines are spoken by a single Character, and may address the other Character (interlocutor) on stage. If a second Character is addressed, but is not on stage, the program throws an error.
- Each line expresses program operations through Sentences. Lines need at least one.

Examples

a)
```
Horatio:
  You lying stupid fatherless big smelly half-witted coward!
```

b)
```
Juliet:
  You are as cowardly as the sum of yourself and the difference
  between a big mighty proud kingdom and a horse.

  Speak your mind!
```

8 Interaction

An interaction line is a command to either open input from the user, or output the value of a character.

Syntax

```
Interaction    ::=    Input | Output
Input          ::=    Listen to your heart ST
                 |    Open your mind ST
Output         ::=    Open your heart ST
                      Speak your mind ST
ST             ::=    . | !
```

Semantics

- Input and Output phrases can only be used with two Characters on stage. An input or output phrase spoken by a Character always references the value of the interlocutor.
- Input denoted by the phrase “Listen to your heart” halts program execution, and takes in a single signed integer.
- Input denoted by the phrase “Open your mind” halts program execution, and takes in a single ASCII character. The character is converted into its corresponding integer value, and assigned to the recipient.
- Output denoted by the phrase “Open your heart” outputs the interlocutor signed integer value, or null;
- “Speak your mind” outputs the ASCII character represented by the interlocutor integer value. If the integer value is null or negative, the output is an empty char.

Examples

a)
```
Horatio:
  Open your mind! Remember thyself.
```

b)
```
Romeo:
  You are the sum of a dim-witted coward and a beautiful hero.
  Speak your mind!
```

9 Goto

Goto statements allow the program to travel to any Scene within the same Act. It can be explicit, or dependent on a conditional statement.

Syntax

Goto	::=	Imperative Return **to** Target **.**
Imperative	::=	**Let us** | **let us** | **We shall** | **we shall** | **We must** | **we must**
Return	::=	**proceed** | **return**
Target	::=	**Scene** Numeral **.**

Semantics

- Imperatives must use one of the assigned values. This initializes the Goto routine.
- Target is denoted by the keyword Scene followed by a roman numeral. The numeral must be of a Scene within the current Part (Act). If it is not, an error is thrown.

Examples

a)
```
Romeo:
  Thou art a smelly pig. Am I smaller than the sum of a brave hero
  and thyself?

Juliet:
  If so, let us proceed to Scene III.
```

b)
```
Hamlet:
  We must return to Scene I.
```

10 Conditionals

Conditionals in SPL provide limited comparison testing between two values, allowing testing only of greater-than, less-than, or equal-to.

Syntax

Conditional	::=	Question**?** Response
Response	::=	**If so,** Goto
Question	::=	Be Comparative Value
Comparative	::=	N-Comparative **than**
	|	P-Comparative **than**
	|	E-Comparative
	|	I-Comparative
N-Comparative	::=	**punier** | **smaller** | **worse**
P-Comparative	::=	**better** | **bigger** | **fresher** | **friendlier** | **nicer** | **jollier**
E-Comparative	::=	**as** Adjective **as**
I-Comparative	::=	**not** N-Comparative | **not** P-Comparative

Semantics

- Conditionals are denoted by a Question statement, immediately followed by a Response action.
- Two Characters are required to be present to use a Conditional. One to pose the question, and the other to provide a response action.
- Question statements must begin with an inverted present indicative conjugation of "be", in the form "Am I", "Are you", or "Art thou".
- The Question Comparative component expresses the comparison between either the speaking Character or the interlocutor, denoted through appropriate "be" conjugation, and an arbitrary value.
- N-Comparative expresses a "less than" relationship (is a < b?).
- P-Comparative expresses a "greater than" relationship (is a > b?).
- E-Comparative tests for equality. (is a == b?).
- I-Comparatives are an inverse of an N or P nested comparison, equivalent to a "!" (!(a >b) ?).
- N and P comparatives must be denoted by their keyword list.
- E comparatives must have an arbitrary Adjective.
- The following Response indicates a Goto action to be taken when the conditional results in "true".

Examples

```
Horatio:
      Are you as beautiful as a warm summer day?
```

Ophelia:

If so, let us proceed to Scene II.

11 Constants

Constants and constant assignments are the “meat” of SPL. These allow all other operations to take place. A constant consists of a positive noun or negative, representing 1 or -1 respectively, prefixed by zero or more adjectives. Each adjective can be thought of as a multiplier of 2. A constant can therefore be thought of as:

$$\textbf{(the noun value) x } 2^{\textbf{(number of adjectives)}}.$$

Syntax

```
Constant     ::=    P-Constant | N-Constant

P-Constant   ::=    P-Noun
              |     Article P-Noun
              |     Article (P-Adjective | E-Adjective)+ P-Noun

N-Constant   ::=    N-Noun
              |     Article N-Noun
              |     Article (N-Adjective | E-Adjective)+ N-Noun
```

Semantics

- Constants denote signed integers, made of either P or N types.
- Constants cannot be assigned to 0.
- Constants are a noun of value 1 or -1, and an arbitrary amount of corresponding adjectives each raising the value by a power of 2.
- P-Constants consist of a positive noun, and 0 or more positive or neutral adjectives prefixing the noun.
- N-Constants consist of a negative noun, and 0 or more negative or neutral adjectives prefixing the noun.
- P and N Constants cannot be declared with a neutral noun. The adjectives prefixing P or N constants must begin with the respective P or N adjective, not a neutral adjective. P and N constants can begin with an article of “a”, “an”, or “the”, which has no impact on the constant value.
- N-Nouns denote a value of -1.

Examples

a) 4 = `beautiful blue summer's day`
b) -8 = `a dirty evil foul toad`
c) 1 = `cousin`
d) -1 = `the leech`

12 Operations

Constants can be manipulated according to basic arithmetic operations, including addition, subtraction, multiplication, and division, as well as a few unary operations.

Syntax

Operation	::=	Unary-Op Value | Arithmetic-Op Value **and** Value
Unary-Op	::=	**the square of**
	|	**the cube of**
	|	**the square root of**
	|	**the factorial of**
	|	**twice**
Arithmetic-Op	::=	**the sum of**
	|	**the difference between**
	|	**the product of**
	|	**the quotient between**
	|	**the remainder of the quotient between**
Value	::=	Operation | Constant | First-Person | Second-Person

Semantics

- Operations are either unary or binary.
- Unary operations take only 1 value.
- Arithmetic (binary) operations take exactly 2 values.
- Operation values can be recursively defined, allowing the ability to nest operations.
- Unary operations passed a value of “null” return a value of “null”.
- Arithmetic operations passed a value of “null” by one or both parameters, return a value of “null”.
- Operation values can be First or Second person pronouns. First person pronouns can be used when the Stage has either one or two Characters present. Second person pronouns can only be used when two Characters are present.

Examples

a) `the difference between a sorry codpiece and a big blue Heaven`

b) `twice the product of thyself and a beautiful angel`

c) `the square of the square root of a bold brave charming hero`

13 Assignments

In order for characters to have values, they must be assigned through dialogue.

Syntax

Assignment	::=	Be A-Value **.**
A-Value	::=	**as** Adjective **as** Value | Value

Semantics

- Assignments must start with a present indicative conjugation of “be” in the form “You are”, “Thou art”, or “I am”. This denotes the target Character of the value assignment.
- An adjective can follow the “be” conjugation, but has no bearing on the value.
- Values can be an Operation, Constant, or a first or second person pronoun.
- First person pronouns can be used with 1 or 2 characters on stage. Second person pronouns must have 2 characters on stage.

Examples

a) `You are as lovely as a foul cowardly frog.`

b) `Thou art the sum of Heaven and Hell.`

c) `I am a half-witted oozing bastard.`

14 Stacks

Characters have stacks of integer values as “memory”, using the basic push/pop-like operations.

Syntax

Stack-Op	::=	Push **!** | Pop **!**
Push	::=	**Remember** Pronoun
Pop	::=	**Recall** Comment

Semantics

- Stack commands require 2 characters on stage. The speaker modifies the interlocutor’s stack.
- The Push command does not modify the interlocutor’s present value.
- The Push command stores the value of the Character indicated through the first or second person pronoun used on the top of their stack.
- The Pop command removes the top value from the interlocutor’s stack, and assigns their value to the popped value.
- If the interlocutor’s stack is empty, the Pop command assigns their value to null.

15 Lexicon

Syntax

Article	::=	**a** | **an** | **the**
Be	::=	**Thou art** | **You are** | **I am**
Pronoun	::=	First-Person | Second-Person
First-Person	::=	**I** | **myself** | **me**
Second-Person	::=	**you** | **yourself** | **thy** | **thyself** | **thee** | **thou**
Adjective	::=	P-Adjective | E-Adjective | N-Adjective
P-Adjective	::=	**amazing** | **beautiful** | **blossoming** | **bold** | **brave** | **charming** | **clearest** | **cunning** | **cute** | **delicious** | **embroidered** | **fair** | **fine** | **gentle** | **golden** | **good** | **handsome** | **happy** | **healthy** | **honest** | **lovely** | **loving** | **mighty** | **noble** | **peaceful** | **pretty** | **prompt** | **proud** | **reddest** | **rich** | **smooth** | **sunny** | **sweet** | **sweetest** | **trustworthy** | **warm**
E-Adjective	::=	**big** | **black** | **blue** | **bluest** | **bottomless** | **furry** | **green** | **hard** | **huge** | **large** | **little** | **normal** | **old** | **purple** | **red** | **rural** | **small** | **tiny** | **white** | **yellow**
N-Adjective	::=	**bad** | **cowardly** | **cursed** | **damned** | **dirty** | **disgusting** | **distasteful** | **dusty** | **evil** | **fat** | **fat-kidneyed** | **fatherless** | **foul** | **hairy** | **half-witted** | **horrible** | **horrid** | **infected** | **lying** | **miserable** | **misused** | **oozing** | **rotten** | **smelly** | **snotty** | **sorry** | **stinking** | **stuffed** | **stupid** | **vile** | **villainous** | **worried**
Noun	::=	P-Noun | N-Noun
P-Noun	::=	E-Noun | **Heaven** | **King** | **Lord** | **angel** | **flower** | **happiness** | **joy** | **plum** | **summer's day** | **hero** | **rose** | **kingdom** | **pony**
E-Noun	::=	**animal** | **aunt** | **brother** | **cat** | **chihuahua** | **cousin** | **cow** | **daughter** | **door** | **face** | **father** | **fellow** | **granddaughter** | **grandfather** | **grandmother** | **grandson** | **hair** | **hamster** | **horse** | **lamp** | **lantern** | **mistletoe** | **moon** | **morning** | **mother** | **nephew** | **niece** | **nose** | **purse** | **road** | **roman** | **sister** | **sky** | **son** | **squirrel** | **stone wall** | **thing** | **town** | **tree** | **uncle** | **wind**
N-Noun	::=	**Hell** | **bastard** | **beggar** | **blister** | **codpiece** | **coward** | **curse** | **death** | **devil** | **drought** | **famine** | **flirt-gill** | **goat** | **hate** | **hog** |

```
                              hound | leech | lie | pig | plague | starvation | toad | war | wolf

Numeral          ::=          Hundreds Tens Units
Hundreds         ::=          ε | C | CC | CCC | CD
Tens             ::=          LowTens | XL | L LowTens | XC
LowTens          ::=          ε | LowTens X
Units            ::=          LowUnits | IV | V LowUnits | IX
LowUnits         ::=          ε | LowUnits I

Comment          ::=          Graphic*
Graphic          ::=          Letter | Digit | Op-character | space | tab | : | ; | , | ~ | ( | ) |
                              [ | ] | { | } | _ | | | ! | ‘ | ` | “ | # | $
Letter           ::=          a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s |
                              t | u | v | w | x | y | z | A | B | C | D | E | F | G | H | I | J | K |
                              L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z
Digit            ::=          0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
Op-character     ::=          + | - | * | / | = | < | > | \ | & | @ | % | ^ | ?
```

Notes

- Roman Numeral syntax borrowed from http://homepage.cs.uiowa.edu/~hzhang/c123/KeyCh09.pdf

References

- http://shakespearelang.sourceforge.net/report/shakespeare/