VENTURE

COMS 4115 - Language Proposal
Zach Adler (zpa2001), Ben Carlin (bc2620), Naina Sahrawat (ns3001), James Sands (js4597)

Description of language we plan to implement:

We plan to build a Java-like object-oriented programming language. We will then build a library using the language which is designed to allow game developers to easily create text-based adventure games.
Text-based adventure games, a style of game that has remained relevant in the gaming world despite the advent of complex and high-technology special effects and multiplayer interfaces, are built mainly for a solo player to interact with a storyline through just text. We propose a language designed to make the process of developing such a game trivial for those without programming experience, while providing more experienced programmers the ability to create more complex games in less time. Character and world building can be done via JSON files rather than hard programming them, and the language will compile into LLVM. Our language provides the essential components of all text-based adventure games right into our language as data types, such as rooms, items, and Non-Player Characters (NPCs). Our proposal is to build Venture, the language used to create worlds.

Types of programs that will be written in our language:

Our language will be used to design and create their own version of the game 'Adventure' with minimal programming required. Objects such as equipment with default attributes which the player can interact with in game can also be created via JSON, or the developer can code their own unique objects. Interactions between the player and an object and between the player and NPCs will be similarly handled by Venture; a set of defaults will be provided for creation via JSON objects and developers can create their own as they desire.
After creating the game world, including NPC's, rooms, items, and their interactions, Venture will compile the provided information into an executable that will allow a user to play the game.

Language Basics:

Primitive	Examples
int	-110, 49, 3, 0
string	"a", "bc", "STRING"
boolean	TRUE, FALSE
LinkedList	String[], Room[]

Linked List generic Methods:

Datatype	Method	Description
void	insert(Object x)	Insert into LinkedList
void	remove(Object x)	Remove from LinkedList
int	size()	Return the number of elements in LinkedList
boolean	contains(Object x)	Return TRUE if x is in LinkedList; FALSE if not

Operator	Description
* / %	multiplication, integer division, modulo
+ -	Addition, subtraction
< <= > =>	Less than, less than or equal to, greater than, greater than or equal to
== !=	equal, unequal
=	assignment
AND	Logical AND
OR	Logical OR

Keyword	Description
if, else	Control flow statements
while	loops
new	Creation of user defined Objects

I/O Library Functions:

Datatype	Method	Description
String	.read()	Reads in user input and returns input as String (for user input, stdin)
String	.readLine()	Reads in input until a new-line character is found, returns that line that was just read (for file use)
String	print(String str)	Prints out String to Console (stdout)

Venture Game Building Library:

Library Datatypes:

Room:

Description: An instance of a room object represents a container that the player can move in and out of. NPCs and items can exist in the container.

Attribute	Datatype	Description
name	String	Name of room
items	String[]	List of items in the room
adjacent_rooms	String[]	List of adjacent rooms
locked	boolean	Indicates if key is needed to open door
active	boolean	Boolean used to mark if the player is currently in the room
description	String	Text displayed when room entered for narrative purposes

Note: Rooms may not have the same name. Will throw exception.

Item:

User defined items (eg. knife, book, shield)

Attribute	Datatype	Description
name	String	Name of item
actions	String[]	List of available actions for item when it is in the player's inventory
weight	int	Weight of the item, to be recorded in the inventory.
armor	int	Items such as shields will automatically prevent damage.
damage	int	Damage of item if used to attack. Would be default 0pts.
description	int	Description of item to be used when interacted with.

Notes:

There will be a default list of recognized actions for actions_inventory (eg. 'attack with', 'drop', 'read') that a game developer can use. These actions will be associated with reactions (eg. attacking an NPC with a knife will initiate combat and reduce their health). We will provide functionality for a developer to code a custom action and the reactions as they see fit.

Items may not have the same name. Will throw exception.

NPC:

Description: NPCs, short for non-playable characters, are the characters that a user can talk/interact with in the game world.

Attribute	Datatype	Description
name	String	Name of NPC
inventory	String[]	List of items the character possesses
health	int	Health of character used for combat.
dialogue	String	Character dialogue is printed upon prompt.
combat	boolean	Boolean used to indicate whether or not the NPC and the player are in active combat.

Notes:

Dialogue in the game will not be interactive. There will only be an option for a player to prompt an NPC at which point the game would print the NPC's dialogue.

Combat boolean set to False for character by default. Can be set to True during character creation in which case the character attacks the player on sight.

NPCs may not have the same name. Will throw exception.

Player:

Description: Players is the object that stores information about the user's avatar in the game such as health, and inventory.

Attribute	Datatype	Description
inventory	String[]	List of items the character possesses
health	Integer	Health of character used for combat.
carry_weight	Integer	Carry weight of character. Character has set weight limit and can only carry a limited number of items.

Task:

Description: Task which the player has to complete

Attribute	Datatype	Description
active	boolean	True if active. False if task completed.
description	String	Description of task.
name	String	Name of task.

Note: Tasks may not have the same name. Will throw exception.

Game:

Description: Game objects which is used to keep track of the rooms, interactions and game history.

Attribute	Datatype	Description
rooms	Room[]	Linked list of room names.
tasks	Task[]	Linked list of names of tasks.
name	String	Name of game

Sample code:

Object building:

~example of basic class creation in Venture~

```
class Node{
	private int node_value;
	~constructor~
	public  Node(int x) {
		node_value = x;
	}
	~function~
	public void change_value(int value) {
		node_value = value;
	}
}
```

Game building:

```
include Venture_library;
public void main(){
```

~initialize knife: item(name, actions, weight, armor, damage, description)~

```
String knife_description = "You pick up the awesome knife. You feel ready to
kill some goblins.";
item knife = new item("sweet knife", ["attack", "throw"], 5, 0, 10,
knife_description);
```

~Create character and room~

~NPC(name, inventory, health, dialogue, combat) ~

```
NPC goblin = new NPC("Goblin King", [],  30, "I'm a Goblin!", True);
```

~room(name, items, adjacent_rooms, locked, active, description)~

```
String entrance_description = "You wake up outside a cave. Your head hurts";
```

~active = True means this is the starting location of the player.~

```
        room entrance = new room("Entrance to Cave", ["sweet knife"], ["Main Cave"],
        False, True, entrance_description);

        String cave_description = "The cave is dark. You hear a growl!";
        room cave = new room("Main Cave", [], ["Entrance to Cave"], False, False,
        cave_description);

        new game cave_adventure([entrance, cave], [], "Goblin Quest" );
}
```

~From the code above using the Venture library, a developer would have completed a game.
In this game, the player wakes up outside the cave. The description of the entrance would print on the screen. The player could then pick up the knife (advisable) and enter the main cave. The description of the main cave would print out, then the Goblin King would say "I'm a goblin" and initiate combat.~