Wang Yuanxi A1805637 - Chan Jing Kai Germin A1805312

PROJECT SPECIFICATION - MAJOR PRACTICAL

INTRODUCTION

Our project implements a system for managing soldiers. In the program, there is an enlistment period, where soldiers' names, identification numbers (NRIC), seniority, ranks, and heights are added. These soldiers can be given various events, such as performing a parade, checking a soldier's pay, changing a soldier's rank, and a simulated encounter between 2 soldiers.

DESIGN DESCRIPTION

Assessment concepts

Memory allocation from stack and heap

- **Pointers:** A double pointer of soldiers will be created for the parade, which will be deleted after the parade.

- **Strings:** Each soldier will have a name and rank, which are represented by strings.

- **Objects:** Soldier (abstract class), man, specialist, officer, general

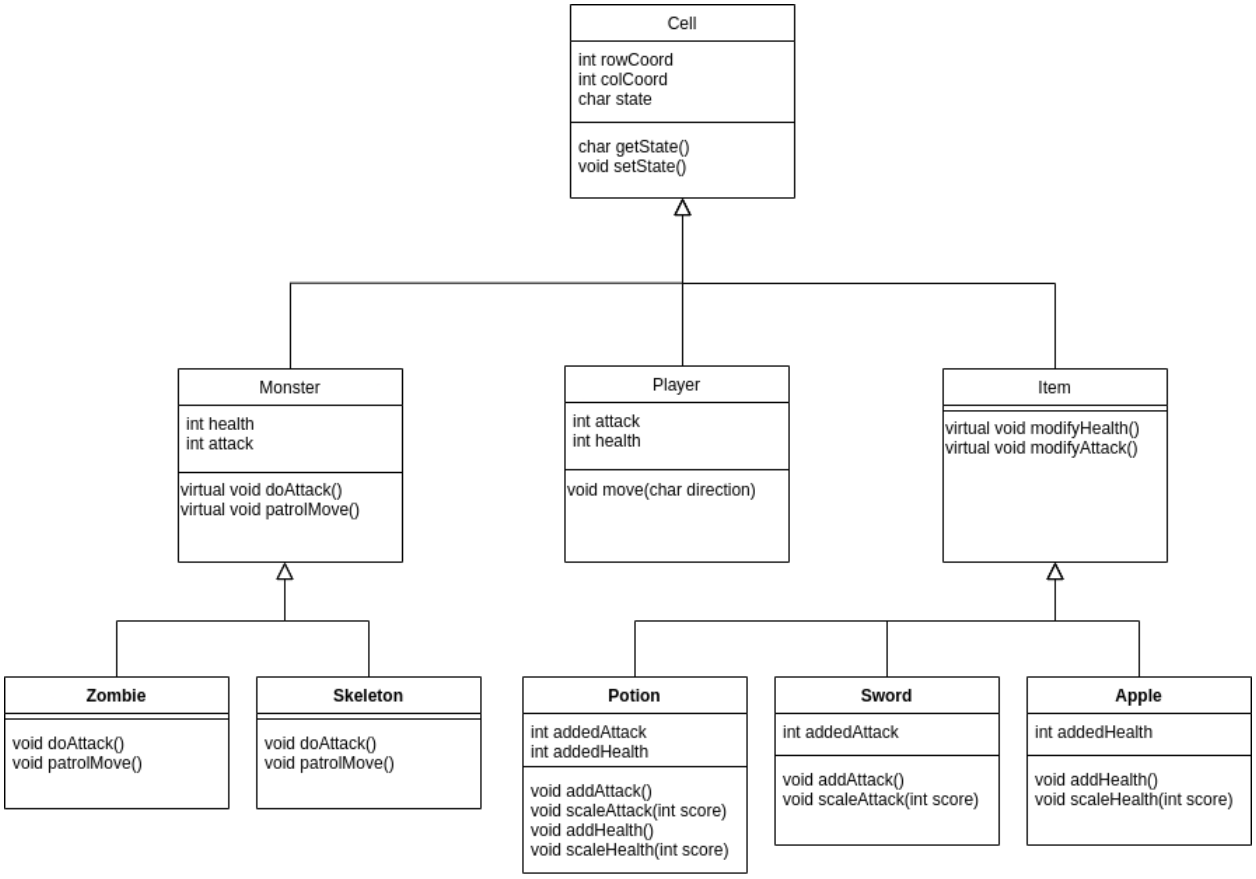
User input and output

- Program uses cin to read in both strings and ints for details of soldiers
- Inputs are in txt files for automated testing

Object-oriented programming and design

- Inheritance: Different ranks of soldiers inherit from soldier, an abstract class

CLASS DIAGRAM



CLASS DESCRIPTIONS

Soldier - abstract class all others inherit from
Man, specialist, officer, general stored in different vectors
Differing pay, ranks, and usage (parade & salute functions)

User interface

Users can enter details of new soldiers and choose what actions to perform with those soldiers

Code Style

All code in the program will be consistently indented using 4 space tabs. Multiple word variables and functions will use camel case. Function Comments will be given for each function that describe what function does.
Comments will be written for code.

TESTING PLAN

The makefile will build the output file, automatically changing objects which have been modified. Testing txt files will compare outputs to expected outputs, to determine unexpected differences.

Unit testing

Each test file will test a different function of the program

- Salute
    - Input seniority and id to find 2 soldiers
    - Simulate an encounter between the soldiers
    - What is likely to be said by first soldier to second soldier
- Parade
    - Generate a parade formation, using all man objects, and 1 each of specialist, officer, and general
- Check Pay
    - Input seniority and id to find specific soldier
    - Output pay of soldier based on their rank
- Change Rank
    - Input seniority and id to find specific soldier
    - promote/demote soldiers to new ranks

Schedule Plan

Stretch Goals

Our goal is to complete most of the functional features by week 10; if we have finished testing by this point, we can expand our program to include more functions to modify other aspects of soldiers, such as changing ids and heights. Also, parade formation creation can take the heights of soldier into account, to generate a formation more similar to real parades.

Week 9

Write Makefile - Germin
Organize code sharing with group members -YX
Create testing strategy - Germin
Create class structure and functionality -YX

Week 10

Create parade and salute functions - YX
Code architect and arrangement and comment for ease of readability - Germin
Create check pay and modify rank functions- Germin

Week 11

Checklist:
Cleanup code (remove unnecessary code) - Germin YX