Concordia University
Department of Computer Science
and Software Engineering

Advanced Programming Practices
SOEN 6441 --- Fall 2011

Project Build 2 Grading

1. Incremental Code Build Description

You must deliver an operational version demonstrating some capacity of your system. This is about demonstrating that the code build is effectively aimed at solving a specific project problem or completely implementing specific system features. The goal of the build must be presented first, then demonstration is made to show that the goal has been met, and also to explain some parts of your code. The code build must not be just a "portion of the final project", but rather be something useful with a purpose on its own.

2. Team Identification

TEAM # :

2. Evaluator Identification

Date	Evaluator	Signature

3. Grading

Presentation		**5**
Explanation of the architectural design of the entire project		1
Knowledge of code base/clarity of explanations		2
Effectiveness and demonstrated preparation of the presentation		2
Character generation/edition		**6**
Creating/editing a *fighter* character following the d20 game rules as described below: level, ability scores and ability modifiers, hit points and base attack modifier proportional to level, armor class adjusted to worn armor and shield.		3
Inventory pane, including worn items: armor, shield, gloves, bracers, rings, helmet, boots, belt, backpack. Equip /unequip items.		2
Save/load a character to/from a file		1
Map generation/edition		**6**
Create a map of a user-defined size		2
Place game elements on a map (entry point, exit point, doors, chests, monsters, …)		3
Save/load a map to/from a file		1
Play		**15**
Select a map and character from a list of saved ones		1
Start the game by having the player character placed on the starting point, the monsters' levels are adjusted to the level of the player character		2
Player character can equip/unequip items during play		2
The game follows the game sequence as described below, including moving and attack following the d20 rules		4
End the game by having the character stepping on the exit point, the exit point being activated only when all monsters on the map have been defeated		1
Player character automatically goes up a level and is saved upon exiting the map		1
Logging window clearly showing that the d20 rules are followed during game play, including dice rolls, and all applicable values and modifiers used in the calculations		4
Use of tools/techniques		**18**
Software versioning repository (history, build 1 and 2 baselines)		3
Inline documentation (each class and method, available on team's web space)		3
Unit testing framework (at least 30 relevant test cases, single test suite)		3
Coding standards (meaningful variable names, appropriate use of comments)		3
Use of design patterns (Character builder, Map builder + 1 other of your choice)		6
Total		**50**

Implement a Builder pattern for the Character class to create characters (player character or enemy character) of various levels (fighter class only), and enabling various types of fighter style to be chosen (see table below).

Ability scores generation method: Any character (and even monsters) has the same 6 ability scores (Strength, Intelligence, Dexterity, Constitution, Wisdom, Charisma). Upon creation of the character, the values associated with each ability score is randomly determined. For the generation of ability scores, for each ability score, roll 4d6 and selects the 3 highest dice values. After all 6 scores have been generated, they are assigned to an ability depending on the type of fighter that this character is: (1) a *"bully"* uses brute strength to destroy his enemies, (2) a *"nimble"* favors dexterity and better armor class to evade blows, (3) a *"tank"* favors survival by more hit points through a high constitution score. Create one Concrete Builder for each of these three types of fighter.

Type of fighter	Ability scores in decreasing order of importance
Bully	Strength, Constitution, Dexterity, Intelligence, Charisma, Wisdom
Nimble	Dexterity, Constitution, Strength, Intelligence, Charisma, Wisdom
Tank	Constitution, Dexterity, Strength, Intelligence, Charisma, Wisdom

Level-dependent characteristics: As a character goes up levels, the following are increasing: (1) his hit points go up by (1d10+constitution modifier), (2) his base attack bonus goes up by one, and his number of attacks per round increase by one every five levels, according to the following table:

level	base att bonus
1	+1
2	+2
3	+3
4	+4
5	+5

level	base att bonus
6	+6/+1
7	+7/+2
8	+8/+3
9	+9/+4
10	+10/+5

level	base att bonus
11	+11/+6/+1
12	+12/+7/+2
13	+13/+8/+3
14	+14/+9/+4
15	+15/+10/+5

level	base att bonus
16	+16/+11/+6/+1
17	+17/+12/+7/+2
18	+18/+13/+8/+3
19	+19/+14/+9/+4
20	+20/+15/+10/+5

The ability modifiers are calculated from the baseline of a value of 0 for an ability value of [10-11] and decrease by 1 for each ability value of 2 less (i.e. [8-9] is -1, [6-7] is -2, ... , [0-1] is -5). As the minimum ability score is 0, the minimum ability modifier is -5. Similarly, the ability modifier increases by 1 for each ability value of 2 more (i.e. [12-13] is +1, [14-15] is +2, [16-17] is +3, etc). As there is no upper limit to the ability scores values, there is no upper limit to the ability modifiers.

The armor class is calculated according to the dexterity modifier and the armor and shield worn by the character: AC = 10 + (dex mod + armor mod + shield mod). The armor modifier depends on the type of armor the character is wearing and ranges from +1 to +8 (see below).

armod mod	armor
1	padded
2	leather
3	studded leather

armod mod	armor
4	chain shirt
5	breast plate
6	banded mail

armod mod	armor
7	half plate
8	full plate

The shield modifier applies if you are wearing a shield and is +1, +2 or +4 depending on the type of shield (see below).

shield mod	shield
1	buckler
2	heavy shield
4	tower shield

The attack bonus is calculated according to the base attack bonus (see above) and the ability modifier (for melee weapons, use the strength ability modifier; for ranged weapons, use the dexterity ability modifier). The attack bonus is used in the combat sequence to determine if a successful hit is happening.

weapon type	attack bonus
melee (e.g longsword)	base attack bonus + strength modifier
ranged (e.g. longbow)	base attack bonus + strength modifier

The game sequence is turn-based. Every turn, each participant (player or monster) gets to do an action. At the beginning of every turn, an initiative roll is made by each participant (1d20 + dexterity modifier). The values of each roll determines the order into which each participant will get his turn. Each participant is doing their allowed action each turn, which may involve movement (i.e. moving a maximum of 6 squares) and/or attacking. When an attack is declared on a target, 1d20 is rolled and added with the attack bonus of the attacker. If the result is greater or equal than the armor class of the defender, the attack results in a hit and hit points are deducted from the defender. Both a longsword and longbow will inflict 1d8 hit points of damage. Melee weapons' damage are adjusted with the strength modifier. Ranged weapons' damage inflict only their nominal damage and is not affected by strength nor dexterity. During an attack, the attacker is allowed a number of attacks proportional to its level, as shown on the table above. Melee attacks are allowed only when the attacker and defender are on adjacent cells. Ranged attacks are allowed from non-adjacent cells.

weapon type	damage	damage bonus
longsword (melee)	1d8	strength modifier
longbow (ranged)	1d8	none

Once a player or monster has reached 0 or less hit points, he is declared dead. Once every participant has done his own action in the turn, a new initiative roll is made and a new turn starts. In order to demonstrate that your combat sequence is implemented properly using the d20 rules, you have to implement a logging window into which the user can clearly follow step-by-step what is happening during the game showing the calculations happening in order to determine successful hits and damage inflicted, showing all dice rolls and modifiers involved in the calculation.

Implement a Builder pattern for the Map class. Such a builder will take any map previously saved by the Map Generator/Editor and adjust the level of the monsters on this map, as well as (eventually) the value of the items found in the chests on this map.

The playing sequence to be demonstrated in this build is the following: (1) select a map and a player character; (2) as the game starts, the map is to be adjusted automatically to the level of the character entering the map, i.e. the monster on this map should be the same level as the player character; (3) upon entering the map, the player character is placed on the entry point; (4) the game follows the turn-based game sequence as described above, including moves and attacks; (5) the player character defeats the monster following the d20 rules; (6) the player character exits the map and automatically goes up a level and is saved automatically.