CSCI3100 Software Engineering

Assignment 1

Due – midnight, 7th Feb, 2021 (Sunday)

Q1 Answer:

a)

Quality	Internal	External	Product	Process
Correctness	X	X	X	X
Reliability		X	X	X
Robustness	X	X	X	X
Performance		X	X	X
User Friendliness		X	X	
Verifiability	X	X	X	X
Maintainability	X		X	X
Repairability	X		X	
Evolvability	X		X	X
Reusability	X		X	X
Portability	X	X	X	
Understandability	X	X	X	
Interoperability	X	X	X	
Productivity	X			X
Timeliness	X	X		X
Visibility	X	X	X	X

b) The answer is open. Here are examples of each of the software quality:
Correctness: A customer wants a "product" to be correct, and it is the "developer's" responsibility to make it correct. Often the "user" can see incorrectness.
Reliability: A customer can see if a product is reliable (consistent).
Robustness: Software should be developed to handle the unexpected (as practical). The user might be able to tell how well a product reacts to an unexpected event.
Performance: A developer needs to be aware of the performance of his software product. A user may complain if the performance (e.g., response time) is slow.

Usability (User Friendliness): A developer is concerned about how "friendly" the product, and the user obviously can tell.
Verifiability: As a "developer" puts together a system (process), he wants to be able to verify results.
Maintainability: A developer wants his software product to be easily modifiable.
Repairability: Same as above, except repairable.
Evolvability: A developer wants his software product to be easily enhanced.
Portability: If a product needs to be portable, the developer needs to strive for that quality. The user will see if the product not portable.
Understandability: A developer wants his software product to be understandable to simplify maintenance as much as possible.
Interoperability: A developer wants his product to be capable of operating with other products, and a user can see if this is/isn't the case.
Productivity: A developer wants his software development process to be productive.
Timeliness: A developer wants his process to result in a timely completion (so does the user).
Visibility: The developer and the user are both very interested in having quality documentation of the product; The visibility of a software process makes it easier to do changes.

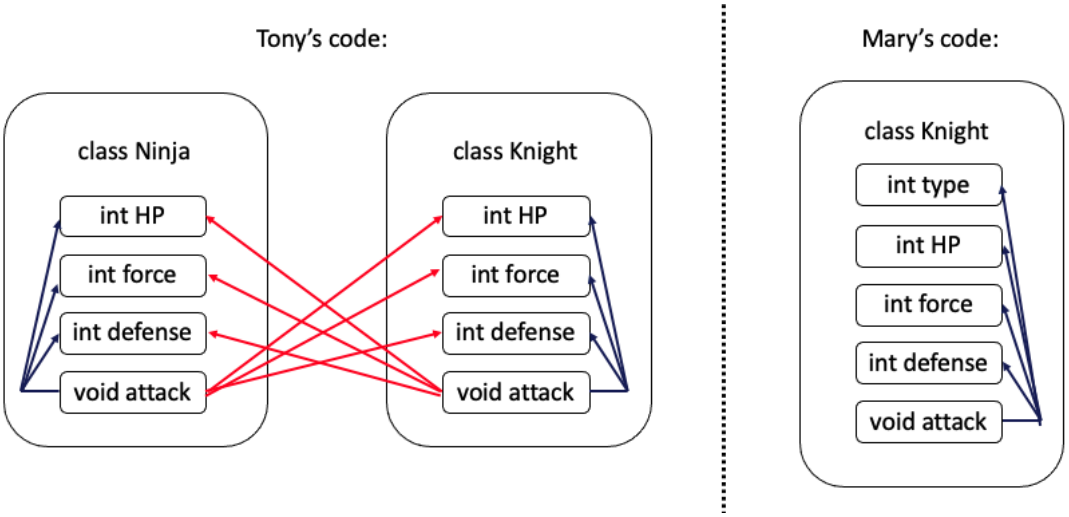
Q2 Answer:

The explanation is very open. Here are some examples:

a) Ture. Maintainability contains repairability and evolvability.
b) True. Correctness is the rigorist quality among dependability qualities.
c) True. If the software is processed efficiently, it will be delivered on time.
d) True. Though the distribution may vary in specified cases or unspecified cases, there is still great chance that they contribute to each other.
e) True. Good performance makes user feel good when using the software.
f) True. If it is easier to understand the software, then the cooperators can easily combine other applications with the software.
g) False. Improving verifiability means it is easier to test the software, but does not directly mean that the software becomes more correct.
h) True. Well documented software can make sure developers understand the organization and structure of the software.
i) False. Usability is for developers; reusability is for users.
j) True. Running in different environments contributes to being reused in different environments.

Q3 Answer:

a) Tony's code: Total cohesion: 10;
Total coupling: 12.
Mary's code: Total cohesion: 16;
Total coupling: 0.
The modularity diagram of Tony's code and Mary's code are as follows (in which blue lines represent cohesion and red lines represent coupling):



Mary's code is better since there are less coupling. This may lead to better maintainability.

b) You will need to add different attack functions in each class, and add several attack functions in class Dragon.
Using inheritance aligns with the principles of Anticipate of Change and Incrementality, since the code can be easier modified when new requirements appear. Also, it aligns with Modularity since a parent class aggregates separate small classes into a unified one.
Using inheritance contributes to Reusability, Usability, Maintainability, Repairability, Evolvability, Interoperability, Portability.

Q4 Answer:

a) The implementation is open. Here is a sample of C-like pseudo code:

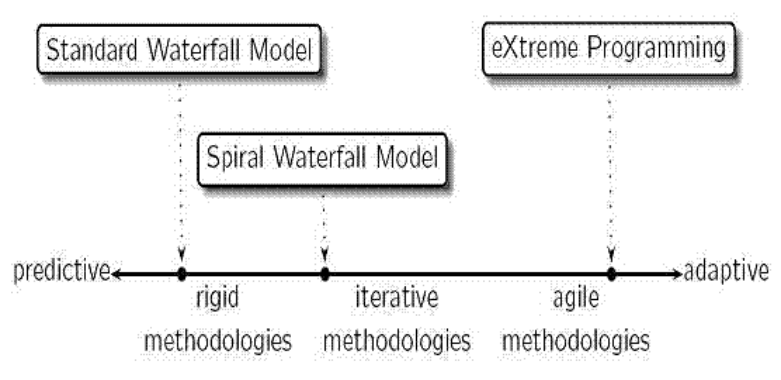
```
class GameController {
    void ProduceSoldiers(){
        /* Produce soldiers according to headquarter.souls_stored */
    };
    void MoveAllSoldiers(){
        /* Maintain the list of soldiers */
    };
    void AttackInCastle(){
        /* Call attack; Update red/blue of a castle */
    };
    void GetSouls(){
        /* Add souls to headquarters according to castle.winner */
    };
    void CheckGameOver(){
        /* Determine whether to end the game */
    };
};
class Castle{
    int winner;
    int souls_stored;
}
class Soldiers{
    int HP;
    int force;
    int defense;
};
class Ninja: public Soldiers{
```

```
28.     void attack();
29. };
30. class Knight: public Soldiers{
31.     void attack();
32. };
33. class Demon: public Soldiers{
34.     void attack();
35. };
```

b) The answer is relatively open.
For example, Fast Prototyping model could be the most proper model.
Because it allows the system to be incrementally improved.
Other software process models are also acceptable, such as Agile model or Extreme programming model.
The following two diagrams show a rough comparison with three chosen Software Development models:

NOTES

A comparison among the standard waterfall model, spiral waterfall model and extreme programming is the following:



Model/Features	Waterfall Model	Extreme Programming	Spiral Model
Requirement Specifications	Beginning	Continuous	High
Simplicity	High	High	Low
Cost	Low	Low	High
Guarantee of success	Low	Low	High
Resource Control	High	Low	High
Cost control	High	Low	High
Risk analysis	High	No risk analysis	High
Expertise required	High	Low	High
User involvement	Only at beginning	Very High	High
Overlapping phases	No	Yes	Yes
Flexibility	Rigid	Very Flexible	Flexible