Week 03 - Object Oriented Principles

Topics covered in this week

- UML
- Composition vs. Inheritance
- Encapsulation
- Polymorphism
- Abstractisation
- SOLID

Materials

- UML: https://www.youtube.com/watch?v=UI6lqHOVHic
- Composition vs Inheritance: https://www.javaworld.com/article/2076814/inheritance-versus-composition--which-one-should-you-choose-.html
- Encapsulation: https://www.tutorialspoint.com/java/java_encapsulation.htm
- Polymorphism: https://www.tutorialspoint.com/java/java_polymorphism.htm
- Abstractisation: https://www.tutorialspoint.com/java/java_abstraction.htm
- SOLID:
    - Articles:
        - http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod
        - http://www.cvc.uab.es/shared/teach/a21291/temes/object_oriented_design/materials_adicionals/principles_and_patterns.pdf
    - Audio/Video:
        - SOLID Principles with Uncle Bob - Robert C. Martin: https://www.hanselminutes.com/145/solid-principles-with-uncle-bob-robert-c-martin
        - Single Responsibility Principle: https://www.youtube.com/watch?v=AEnePs2Evg0
        - Open-Closed Principle: https://www.youtube.com/watch?v=DJF_sGOs2V4
        - Liskov's Substitution Principle: https://www.youtube.com/watch?v=ObHQHszbIcE
        - Interface Segregation Principle: https://www.youtube.com/watch?v=xahwVmf8itI
        - Dependency Inversion Principle: https://www.youtube.com/watch?v=S9awxA1wNNY

Homework

Difficulty	Problem

MEDIUM

The story

- You are making a role playing game.
- First, you'll need a ***hero*** whose role to play. As in all RPGs the hero has health points (life), attack strength and defense. He also has experience points, and a name, given at birth.
- No RPG is fun with the hero only wandering around. They need some ***monsters*** to fight. Not all monsters are the same, some hit you harder, others take more damage at once (read: they have variable attack strength and defense).
- While mosters have no name, they all belong to a specific race that is known. They also have no experience points (since they are meant to be defeated). Different monster races have different magical skills which may enhance their attack or defense.

Your task

- What ***actions are common*** to the hero and the monsters? Group them together!
- Make a Game class with a main method to test your characters. Make them attack each other, and see how they lose HP! For simplicity, you may consider that on each attack the target loses an amount of HP equal to the attacker's strength.

Sample output:

- Created hero Chuck. HP = 100, ATK = 10
- Created monster of race Rock skin cyclops. HP = 30, ATK = 8
- Created monster of race Fire wizard. HP = 50, ATK = 6
- Chuck attacks Rock skin cyclops
-   Chuck's attributes: hp = 100, xp = 10
-   Rock skin cyclops's attributes: hp = 20
- Rock skin cyclops starts using magic
- Chuck attacks Rock skin cyclops
-   Chuck's attributes: hp = 100, xp = 20
-   Rock skin cyclops's attributes: hp = 15
- Fire wizard attacks Chuck
-   Fire wizard's attributes: hp = 50
-   Chuck's attributes: hp = 94, xp = 20
- Chuck attacks Fire wizard
-   Chuck's attributes: hp = 94, xp = 30

Fire wizard's attributes: hp = 40

EASY

Draw an UML class diagram showing the relationships for these entities: music player, CD player, iPod, band, guitarist, drummer, drumset, guitar, lead guitarist, rhythm guitarist, bass guitarist, song, playlist.

- The diagram is correct from the UML point-of-view (for example, the connectors)
- All requested classes are present

You can use https://draw.io (see UML section in the selection pane on the left on this site); export as PNG and put the PNG in the PullRequest on GIT

EASY

Translate the diagram into source code for the previous problem (the music problem)

- All classes from the diagram match the classes from the code
- OOP principles are respected
- Create a CD Player and add a song to it, use as many of the classes defined as needed (try to use all of them if possible)