Assignment #9

Make a battle!

Introduction to Computers II

Tasks

- Modify `NovicePlayer` and its derived classes
- Modify `BaseMonster` and its derived classes
- Design `Battle` class
- Collect similar classes into a directory

Modify `NovicePlayer`

- Add a `protected` variable
  - `int money;` // represents the number of money, >=0
- Add `public` methods
  - `void setMoney();`
  - `int getMoney() const;`

Modify `BaseMonster`

- Add a `public` variable
  - `const int money;` // represents the number of money that it drops after it died

`Battle` class

- Controls a battle between players and monsters
- Features
  - It is turn-based
  - Record number of turns
  - Calculate elapsed turns after a particular action
  - Form a multi-player versus multi-monster battle
  - Display player and monster information
  - Determine which team wins
  - … (whatever you want)

Turn-based: Types

- Entity-scale
  - P1 → E1 → P2 → E2 → P1 → E3 → P2 → E1 → P1 → E2 → P2 → …
- Team-scale
  - P1 → P2 → E1 → E2 → E3 → P1 → P2 → E1 → E2 → E3 → P1 → …

- P: Player
- E: Enemy

Turn-based: Implementation

```
struct Character {
    char type; // monster or player?
    bool alive; // alive or dead?
    void *instance; // pointer to instance
}
```

- A `void` pointer can point(convert) to any type of variables/instances
  - Sometimes it is called generic pointer as well

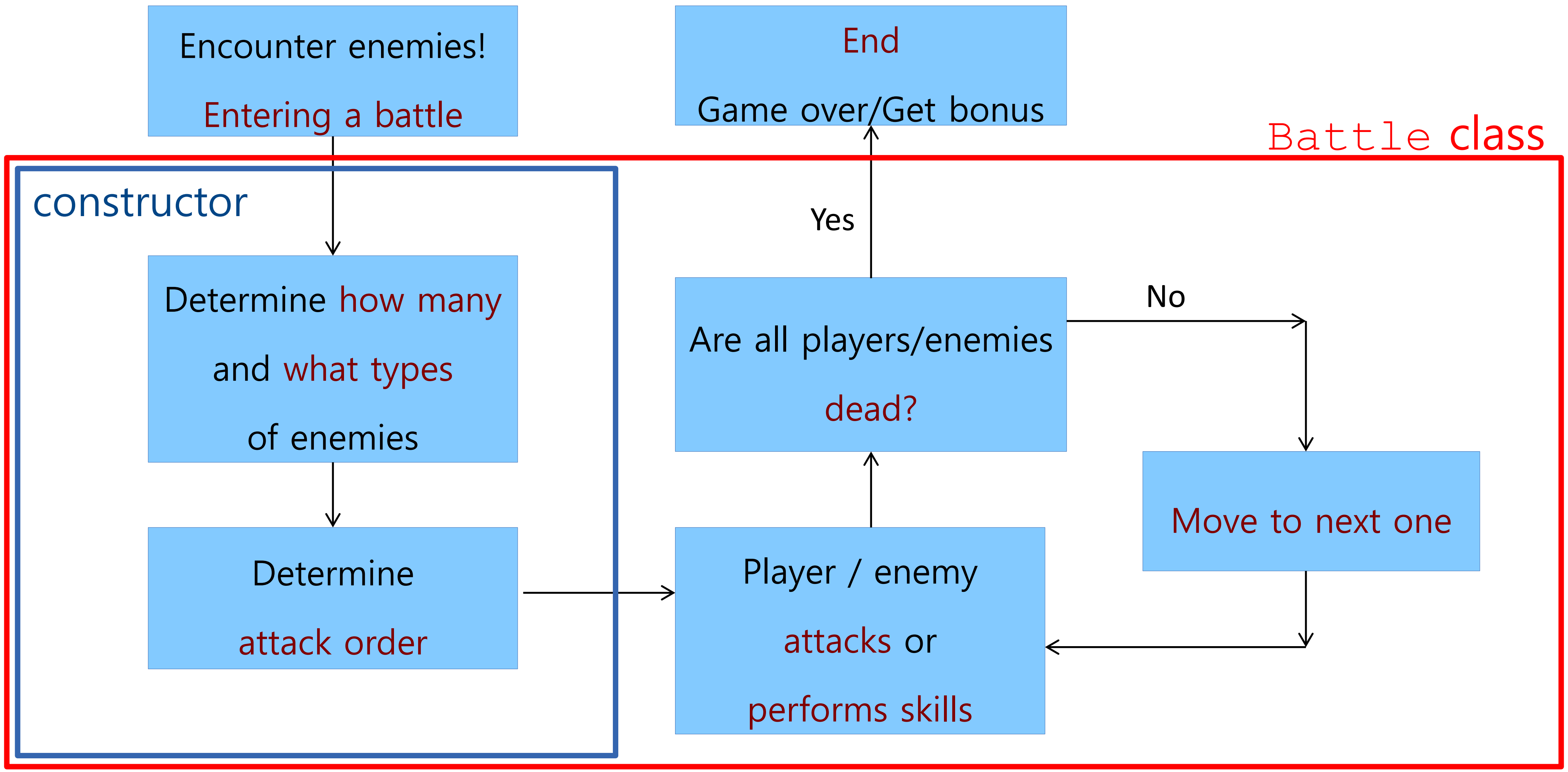
Turn-based: Implementation

```
class Battle {
    private:
        Character *ActionList;
        …
};


Battle::Battle(int nPlyr, int nMon) {
    ActionList = new Character[nPlyr + nMon];
    …
}
```

Turn-based: Implementation

```
NovicePlayer *tmp_player;
BaseMonster *tmp_monster;


if( ActionList[nTurn].type == 'p' ) {
        tmp_player = static_cast<NovicePlayer*>(ActionList[nTurn].instance);
    ...
}
else if( ActionList[nTurn].type == 'm' ) {
        tmp_monster = static_cast<BaseMonster*>(ActionList[nTurn].instance);
    ...
}
```

Revisiting Assignment #9



Collect similar classes

- ./players
  - NovicePlayer.h
  - NovicePlayer.cpp
  - ..
- ./monsters
  - BaseMonster.h
  - BaseMonster.cpp
  - …
- ./misc
  - Battle.h
  - Battle.cpp
  - …

Deliverables

- All class headers and implementations
  - Players: 8 files
  - Monsters: 8 files
  - Battle.h
  - Battle.cpp
- 18 files in total

- Please compress them into a zip archive then upload it to Moodle