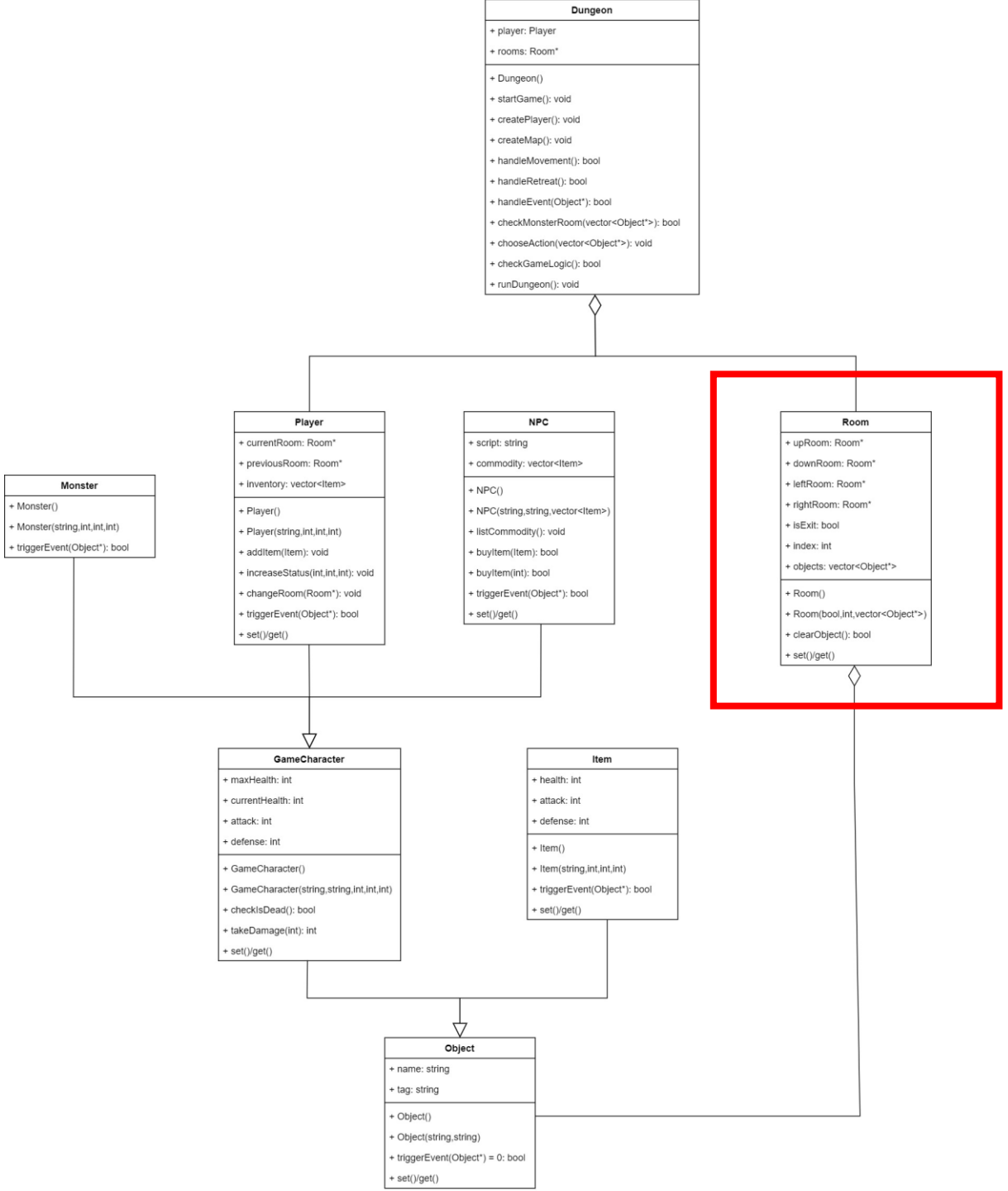
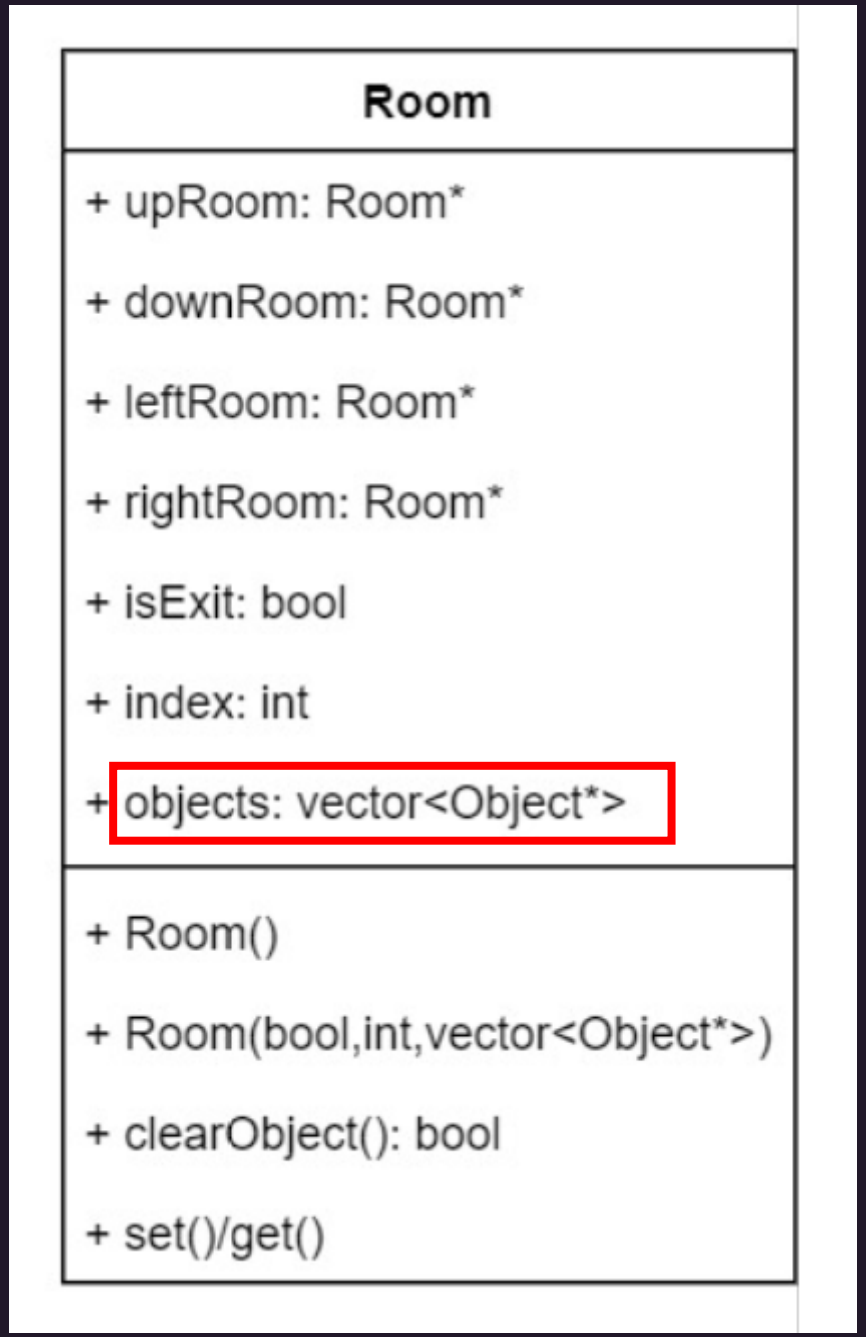
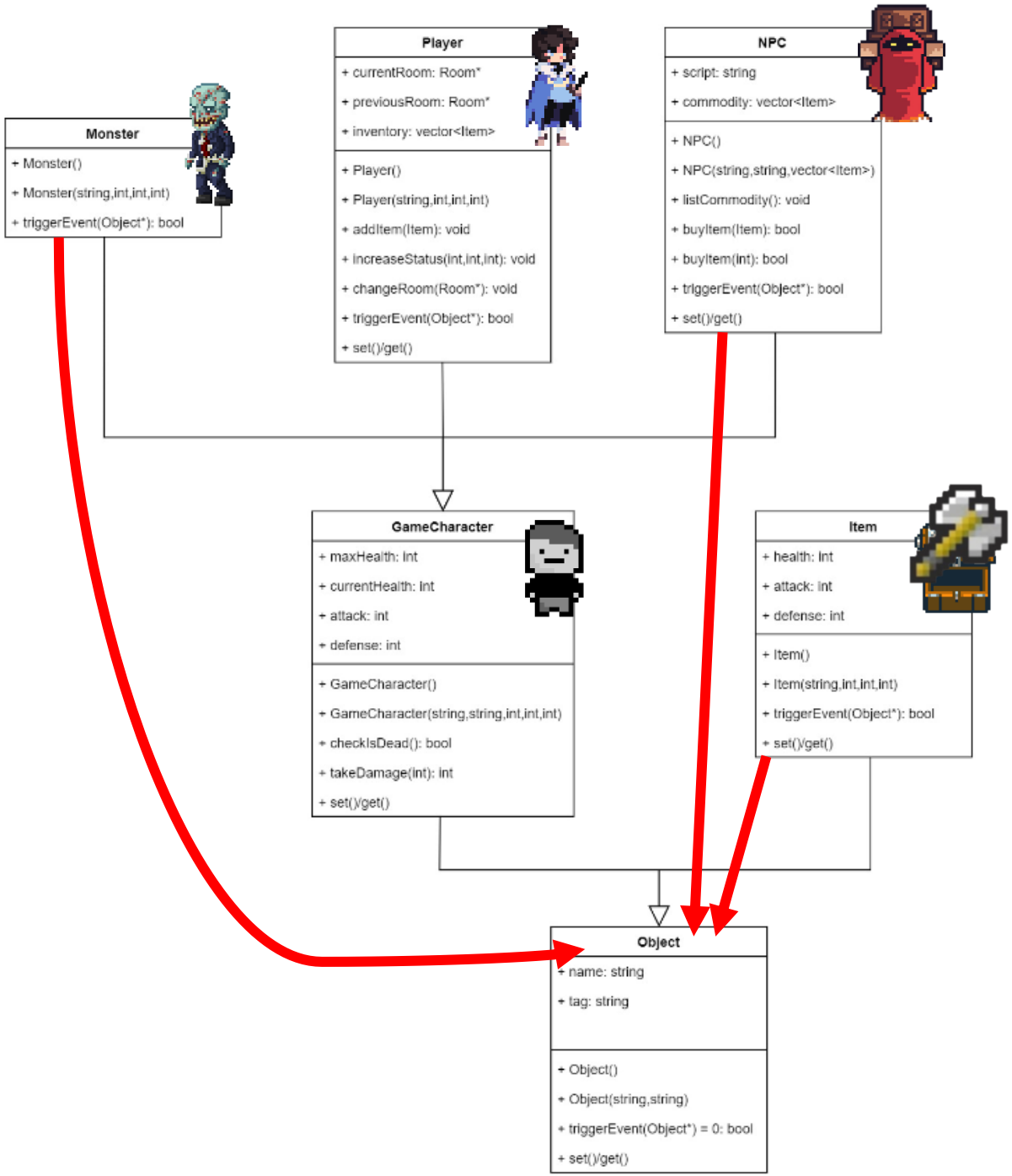
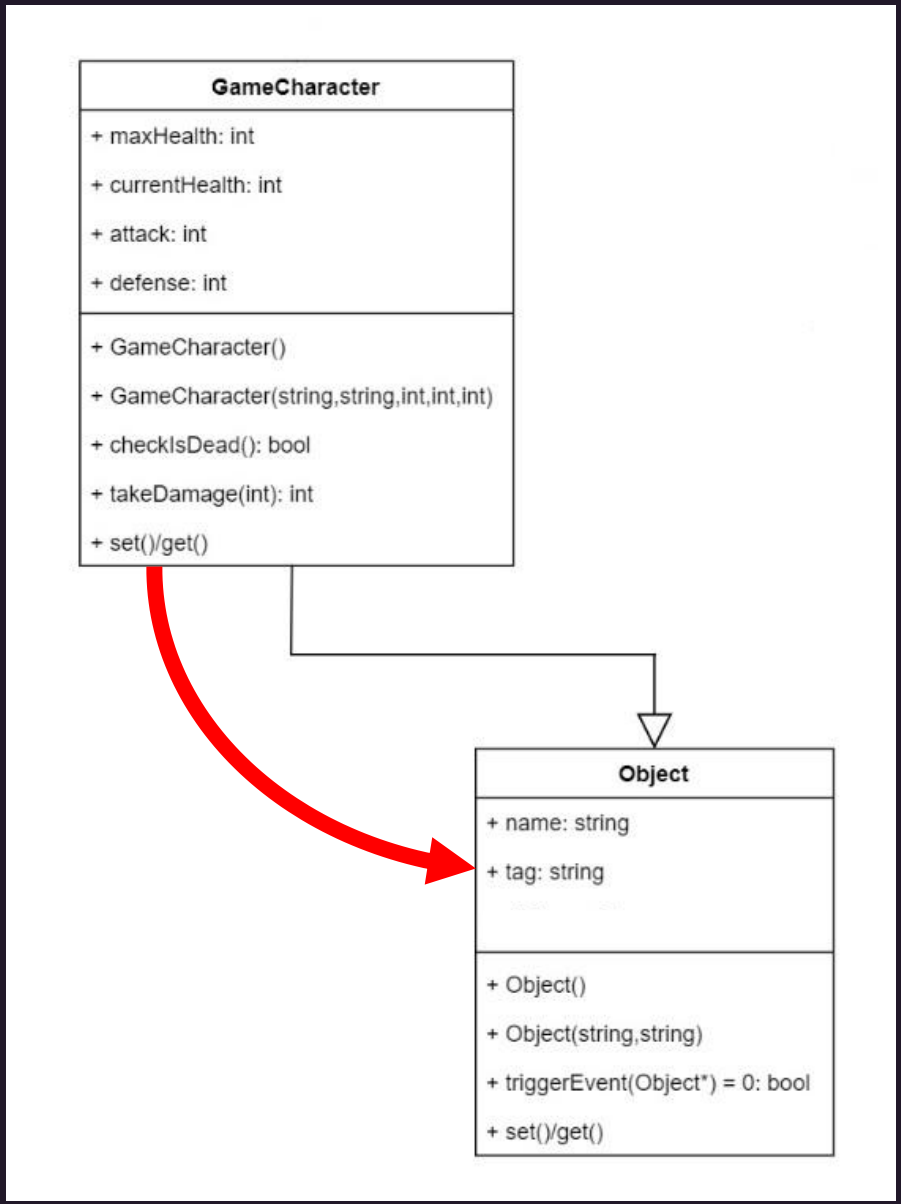
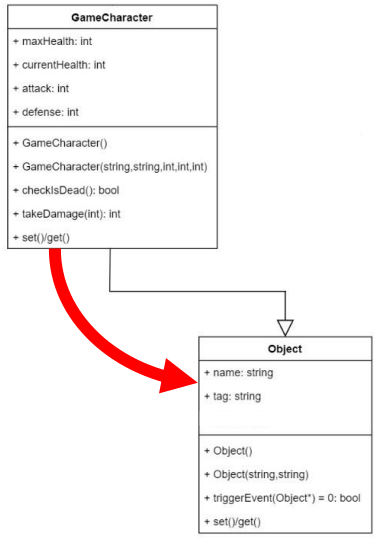
Dungeon


Dungeon
+ player: Player
+ rooms: Room*
+ Dungeon()
+ startGame(): void
+ createPlayer(): void
+ createMap(): void
+ handleMovement(): bool
+ handleRetreat(): bool
+ handleEvent(Object*): bool
+ checkMonsterRoom(vector<Object*>): bool
+ chooseAction(vector<Object*>): void
+ checkGameLogic(): bool
+ runDungeon(): void
Player
+ currentRoom: Room*
+ previousRoom: Room*
+ inventory: vector<Item>
+ Player()
+ Player(string,int,int,int)
+ addItem(Item): void
+ increaseStatus(int,int,int): void
+ changeRoom(Room*): void
+ triggerEvent(Object*): bool
+ set()/get()
NPC
+ script: string
+ commodity: vector<Item>
+ NPC()
+ NPC(string,string,vector<Item>)
+ listCommodity(): void
+ buyItem(Item): bool
+ buyItem(int): bool
+ triggerEvent(Object*): bool
+ set()/get()
Room
+ upRoom: Room*
+ downRoom: Room*
+ leftRoom: Room*
+ rightRoom: Room*
+ isExit: bool
+ index: int
+ objects: vector<Object*>
+ Room()
+ Room(bool,int,vector<Object*>)
+ clearObject(): bool
+ set()/get()
Monster
+ Monster()
+ Monster(string,int,int,int)
+ triggerEvent(Object*): bool
GameCharacter
+ maxHealth: int
+ currentHealth: int
+ attack: int
+ defense: int
+ GameCharacter()
+ GameCharacter(string,string,int,int,int)
+ checkIsDead(): bool
+ takeDamage(int): int
+ set()/get()
Item
+ health: int
+ attack: int
+ defense: int
+ Item()
+ Item(string,int,int,int)
+ triggerEvent(Object*): bool
+ set()/get()
Object
+ name: string
+ tag: string
+ Object()
+ Object(string,string)
+ triggerEvent(Object*) = 0: bool
+ set()/get()


Room
+ upRoom: Room*
+ downRoom: Room*
+ leftRoom: Room*
+ rightRoom: Room*
+ isExit: bool
+ index: int
+ objects: vector<Object*>
+ Room()
+ Room(bool,int,vector<Object*>)
+ clearObject(): bool
+ set()/get()


Player
+ currentRoom: Room*
+ previousRoom: Room*
+ inventory: vector<Item>
+ Player()
+ Player(string,int,int,int)
+ addItem(Item): void
+ increaseStatus(int,int,int): void
+ changeRoom(Room*): void
+ triggerEvent(Object*): bool
+ set()/get()
NPC
+ script: string
+ commodity: vector<Item>
+ NPC()
+ NPC(string,string,vector<Item>)
+ listCommodity(): void
+ buyItem(Item): bool
+ buyItem(int): bool
+ triggerEvent(Object*): bool
+ set()/get()
Monster
+ Monster()
+ Monster(string,int,int,int)
+ triggerEvent(Object*): bool
GameCharacter
+ maxHealth: int
+ currentHealth: int
+ attack: int
+ defense: int
+ GameCharacter()
+ GameCharacter(string,string,int,int,int)
+ checkIsDead(): bool
+ takeDamage(int): int
+ set()/get()
Item
+ health: int
+ attack: int
+ defense: int
+ Item()
+ Item(string,int,int,int)
+ triggerEvent(Object*): bool
+ set()/get()
Object
+ name: string
+ tag: string
+ Object()
+ Object(string,string)
+ triggerEvent(Object*) = 0: bool
+ set()/get()


GameCharacter
+ maxHealth: int
+ currentHealth: int
+ attack: int
+ defense: int
+ GameCharacter()
+ GameCharacter(string,string,int,int,int)
+ checkIsDead(): bool
+ takeDamage(int): int
+ set()/get()
Object
+ name: string
+ tag: string
+ Object()
+ Object(string,string)
+ triggerEvent(Object*) = 0: bool
+ set()/get()




```
41      GameCharactor* charactor = new GameCharactor();
```

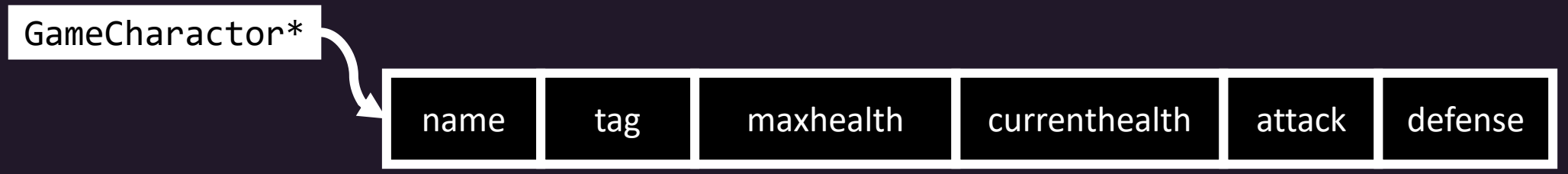
GameCharactor*

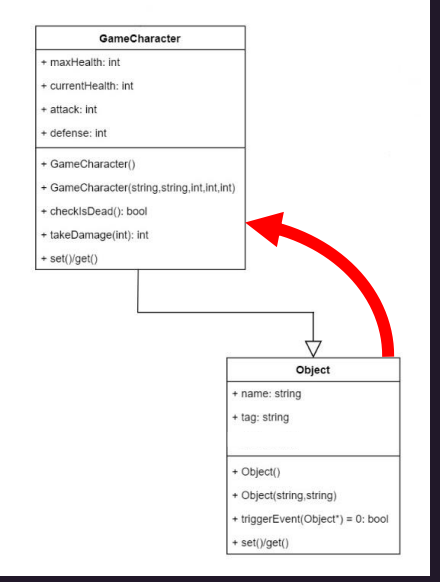

```
43      Object* obj = (Object*) charactor;
```

Object*

```
43        Object* obj = (Object*) charactor;
```

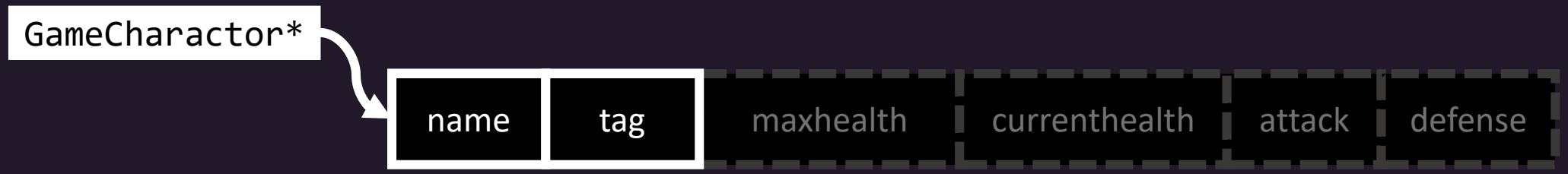

```
45        GameCharactor* role = (GameCharactor*) obj;
```



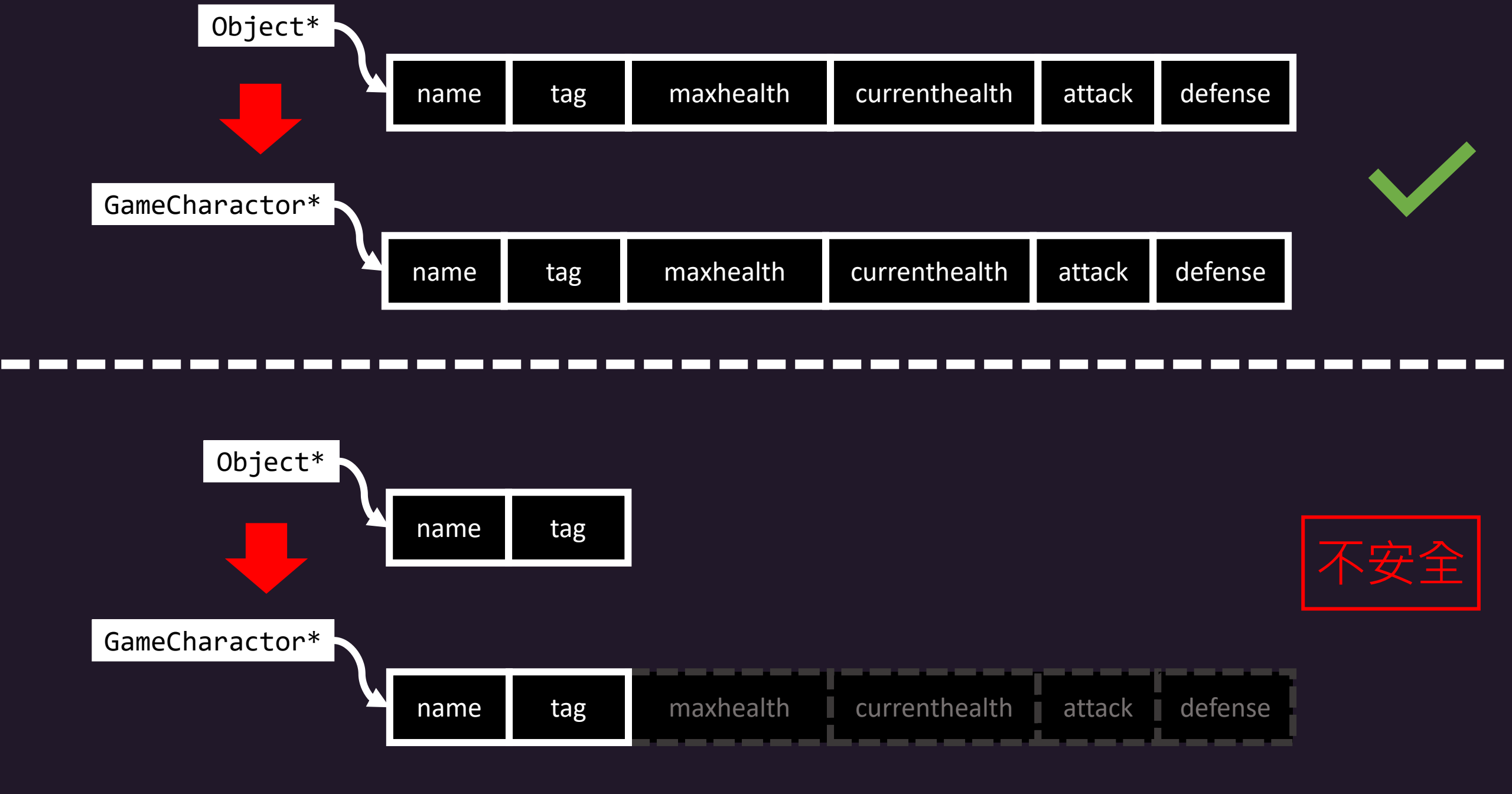



```
45        Object* obj = new Object();
```


```
47        GameCharactor* charactor = (GameCharactor*) obj;
```



不安全的轉換


Object*
name
tag
maxhealth
currenthealth
attack
defense
GameCharactor*
name
tag
maxhealth
currenthealth
attack
defense
Object*
name
tag
不安全
GameCharactor*
name
tag
maxhealth
currenthealth
attack
defense

無型別檢查，直接進行型別轉換

```
41  GameCharactor* charactor = new GameCharactor();
42  Object* obj = (Object*) charactor;
```

有型別檢查，若有問題則回傳空指標

```
44  GameCharactor* charactor = new GameCharactor();
45  Object* obj = dynamic_cast<Object*>(charactor);
```

```
40      Object* obj = new Object();
41      //GameCharactor* charactor = new GameCharactor();
42      GameCharactor* noTypeCheck = (GameCharactor*) obj;
43      GameCharactor* TypeCheck = dynamic_cast<GameCharactor*>(obj);
44      cout << "   obj's addr: " << obj << endl;
45      cout << "no Type Check: " << noTypeCheck << endl;
46      cout << "   Type Check: " << TypeCheck << endl;
```

```
   obj's addr: 0xed1be8
no Type Check: 0xed1be8
   Type Check: 0
```

1. **子類轉父類**(ex GameCharactor -> Object)

   不會有問題，可以直接轉

2. **父類轉子類**(ex Object -> GameCharactor)

   須注意不要存取不存在的屬性

3. **若不確定，可使用dynamic_cast**