Java Dodge 요구 사항

1. [장면]이 있어 장면 마다 다양한 일을 할 수 있다.
   가. 장면: <첫 화면>, <실제 게임>, <게임 오버>, <점수 판>

2. <첫 화면>
   가. 게임 타이틀 표시
   나. 게임 방법 설명
   다. 게임 시작 방법 설명
   라. <점수판> 장면으로 전환되는 버튼
   마. 모든 기록을 지우는 버튼

3. <실제 게임>
   가. 유저가 조종하는 우주선(Ship), 총알들(Bullet), 총알충돌효과(BulletBump), 점수아이템(Gold), 수리아이템(Repair), 체력바(HpBar)가 등장한다.

      1) Ship
         가) 방향키로 움직인다.
         나) 쉬프트키로 속도를 일시적으로 높일 수 있다.
         다) Hull Point(Hp)가 있고 0이하가 되면 <게임 오버> 장면으로 전환된다.
         라) 점수(Score)를 가진다.
         마) Bullet과의 충돌을 체크하여 충돌 시 Hp를 Bullet의 데미지만큼 감소시킨다.
         바) Repair와의 충돌을 체크하여 충돌 시 Hp를 Repair의 값 만큼 증가시킨다.
         사) Gold와의 충돌을 체크하여 충돌 시 Score를 Gold 값 만큼 증가시킨다.

      2) Bullet
         가) 피해량(Damage)을 가져서 Ship과 충돌시에 Hp 계산에 이용된다.
         나) 시간에 따라 화면에 등장하는 양이 증가한다
         다) 시간에 따라 Damage가 증가한다.
         라) Ship에 충돌하면 화면에서 보이지 않는 무작위 좌표로 이동한다.

      3) BulletBump
         가) Bullet과 Ship이 충돌한 장소에 옮겨져서 재생된다.
         나)

      4) Gold
         가) 20개가 화면에 등장한다
         나) 시간에 따라 점수량(value)이 증가한다.
         다) Ship에 충돌하면 화면에서 보이지 않는 무작위 좌표로 이동한다.

      5) Repair
         가) 단 하나만 화면에 등장한다
         나) 시간에 따라 회복량(value)이 증가한다.
         다) Ship에 충돌하면 화면에서 보이지 않는 무작위 좌표로 이동한다.

      6) HpBar
         가) Ship의 Hull Point를 최대 체력의 비율에 따라 표시한다.

4. <게임 오버>
   가. Game Over, 등수, 이름 입력칸(nameBox), 이름 입력 안내(nameInfo), 하이스코어버튼, time, score, result, 우주선 폭발(ShipExplosion)

      1) 엔터를 누르면 게임을 재시작한다(장면 <실제 게임>으로 전환)

      2) 장면 전환시 마지막에 Ship이 있던 장소에 ShipExplosion을 이동하고 재생시킨다.

      3) 등수
         가) 이전까지의 저장된 사용자들의 점수와 비교한 등수를 표시한다
         나) 1등은 YELLOW, 2등은 WHITE, 3등은 GRAY, 4등 이후는 DARK_GRAY

      4) nameBox
         가) 이름을 입력하고 엔터를 누르면 그 이름으로 점수를 저장할 수 있다.
         나) 이름이 입력된 상태에서 엔터를 누르면 그 이름으로 점수를 저장할 수 있다.

      5) 하이스코어버튼
         가) 누르면 현재 nameBox의 이름으로 결과가 저장되고 <점수 판>으로 장면이 전환된다.

      6) time, score, result
         가) <실제 게임>에서 얻은 정보들을 표시한다
         나) result는 time, score에서 산출한다.

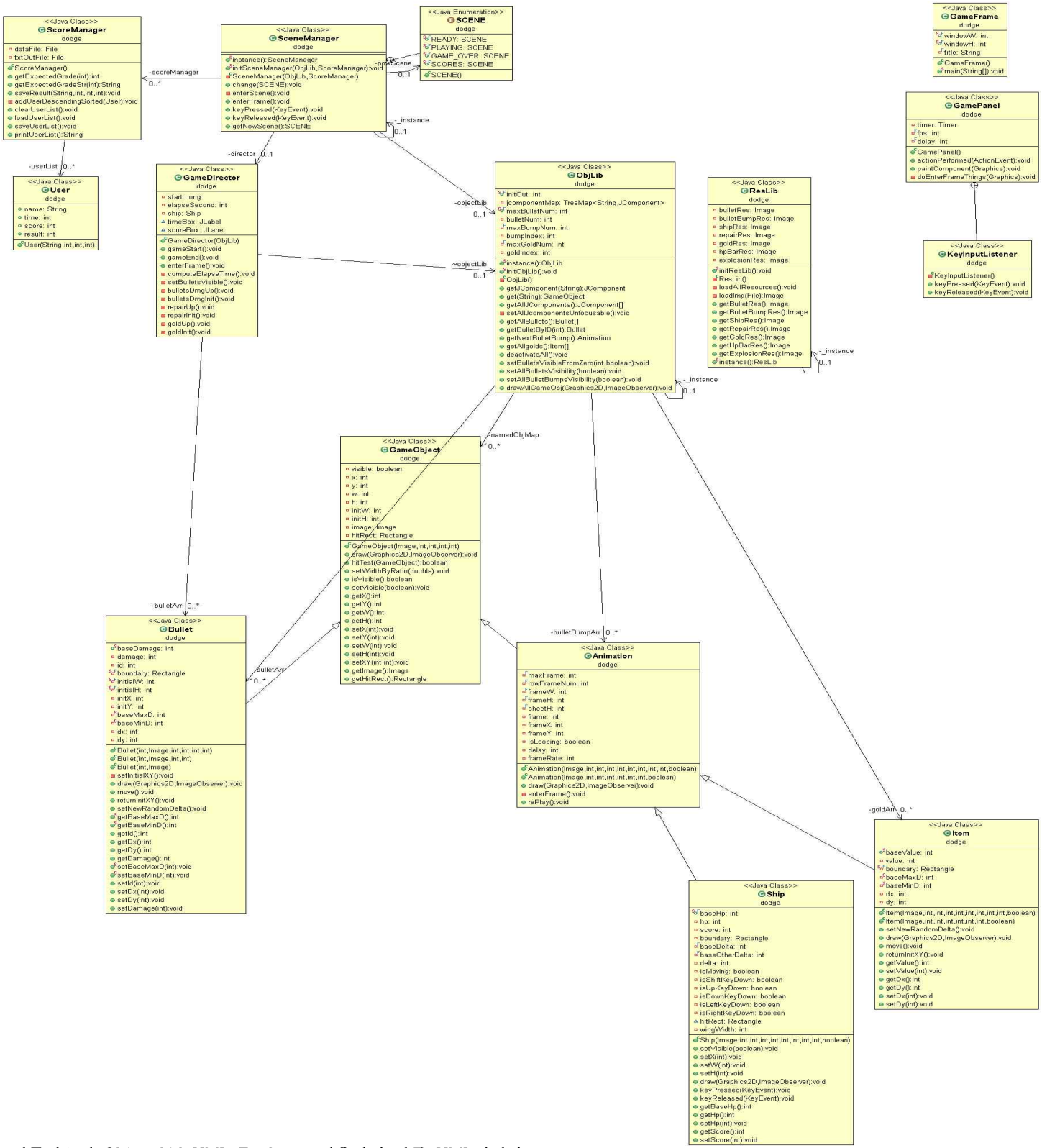
5. <점수 판>

가. 감사인사, ScoreBoard, 크레딧, 엔터키

1) 엔터를 누르면 게임을 재시작한다(장면 <실제 게임>으로 전환)

2) ScoreBoard
가) 파일에 저장된 사용자들의 점수를 일정한 형식으로 출력한다.

6. 기타
가. 게임 시작시에 저장된 사용자들의 점수 목록(UserList)을 로드한다
나. 창이 닫힐 때 UserList를
1) 직렬화하여 저장
2) txt파일로 정해진 형식으로 출력.



-이클립스의 ObjectAid UML Explorer 이용하여 만든 UML입니다.
-원본을 첨부했습니다. 참고하십시오.

게임 구조 요약

1.클래스의 역할

GameFrame	게임의 최상위 컨테이너인 창을 구현한다.
GamePanel	GameObject와 JComponent들이 올려져서 렌더링되는 JPanel이다.
GameObject	게임상에서 렌더링될 수 있는 객체를 구현한다.
Bullet	게임의 적인 총알을 구현한다.
Animation	게임 내 애니메이션 기능을 구현한다.
Ship	사용자가 컨트롤할 수 있는 우주선을 구현한다.
Item	게임상의 아이템인 gold와 repair를 구현한다.
ObjLib	게임에 등장하는 모든 GameObject와 JComponent들을 저장하고 관리한다. (ObjectLibrary)
ResLib	게임에 사용될 모든 리소스를 로드하고 저장한다. (ResourceLibrary)
SceneManager	게임의 장면을 구현하고 관리한다.
GameDirector	<실제 게임>에서 초기화/밸런스를 담당한다.
ScoreManager	사용자들의 점수를 저장하고 관리한다.
User	사용자 한 사람의 점수를 저장하는 단순 자료구조이다.

2.데이터의 흐름

리소스(png파일) -> Image객체(in ResLib)	1:1 대응 관계
Image객체(in ResLib) -> 다양한 GameObject(in ObjLib)	1:다 대응 관계

3.게임 구현 핵심

ObjLib에서 모든 게임 내 렌더링 객체를 생성한 다음 GamePanel에 추가하고, 각각의 <장면>에서 정해진 객체만 활성화(visiblity <- true)시킨다. 장면 전환시에 모든 객체를 비활성시켰다가 다시 장면에 들어가면서 활성화시킨다. 게임 객체들간의 복잡한 상호작용이 있는 <실제 게임> 장면의 상호작용은 GameDirector가 특별히 관리한다.

4.기타

게임 리소스는 Ship 폭발 스프라이트를 제외하고 모두 직접 만들었습니다.
Ship 폭발 스프라이트 출처: http://opengameart.org/sites/default/files/styles/watermarked/public/exp2.png

개선점

이 게임은 약 사흘만에 급하게 만들었기에 코드에 구조적으로 많은 문제점이 있습니다. 시간을 넉넉히 잡고 개발했다면 구조도 생각하고 확장성도 고려하면서 좀더 여유롭게 구현할 수 있었겠지만 이번 학기는 너무나도 바빴기에 불가능했습니다. 아쉬움이 남지만 그래도 다양한 문제점에 대한 개선사항들과 해결방안을 생각해 보았습니다.

무시된 수많은 예외 상황 처리
    싱글톤 패턴이 적용된 객체들의 생성이 단 한 번만 일어나게 강제하기.
    등등..

게임의 정적인 부분(런타임에 바뀌지 않는 부분)과 동적인 부분을 확실히 하고 미흡한 부분을 개선할 것.
    아이템 클래스의 move메서드 알고리즘이 정적인 문제
    전체적인 코드 정리, 문서화. 메서드 주석 달기.

싱글톤 클래스들 캡슐화
    현재 코드는 전역에서 ObjLib에 접근할 수 있음
    ObjLib에 대한 접근을 제한하고 특정 클래스들에게만 허용해야함: 패키지 제한을 위한 default 접근 제한자가 사용되어야 할것임.

메타데이터 이용하기
    ResLib에서 하드코딩된 리소스 로드 코드, ObjLib에서 하드코딩된 객체생성코드, SceneManager에서 하드 코딩된 객체 visibility 설정코드 등을 따로 구조화된 파일과 인터페이스로 관리할 것.

draw와 move의 분리:렌더링과 게잉 상태변경의 분리
    jcomponent가 호출하는 repaint에 의한 버그 픽스
    무의미한 계산(게임 상태 업데이트 횟수 < 렌더링 횟수)이 줄어들 수 있어 성능 향상
    현재는 (게임 상태 업데이트 횟수 = 렌더링 횟수) 이런 코드임.

모든 게임 오브젝트가 준수해야하는 인터페이스 설정
    jcomponent를 상속하는 동시에 위 인터페이스 준수하는 전용 xxx를 만들것.
    ObjLib에 넣을 때, 뺄 때, 모든 게임 객체에 일괄적인 작업(visiblity 조정 등) 시 사용됨

프레임에 따라 움직일 수 있는 객체를 재설계하기
    현재 많은 부분이 겹치는 Item 클래스, bullet 클래스를 리팩토링
    IMovable 같은 걸 만들거나 아예 스프라이트/애니메이션이 공통 상속하는 클래스 만들기

프레임에 따라 움직일 수 있는 객체 move 함수의 동적인(실행시간) 변경
    현재 gold객체들과 repair 객체는 동일한 움직임 알고리즘이 강제되며 런타임 수정도 불가능
    move의 알고리즘을 캡슐화하라. stratage 패턴을 쓸 것

씬노드 구조 도입(SceneManager 리팩토링)
    swirch ~ case의 경직된 구조 해소
    더 쉬운 장면 추가/삭제
    노드가 Scene이 되고, 노드들의 컨테이너가 SceneManager가 된다.

ObjLib와 Scene에서 쓰이는 맵의 키를 문자열 말고 다른 거 쓰기
    지금은 key를 잘못 넣으면 게임이 죽어버린다..