Gebruikte patterns:

Factory pattern – ButtonFactory (knoppen aanmaken) , EntityFactory (entities aanmaken), gebruikt om makkelijker bepaalde zaken aan te maken / genereren.

Singleton – Hero, SoundManager, BossManager, PlayerManager, HUD…. gebruikt om enkel 1 instantie van een bepaalde klasse te hebben in mijn game.

Observer pattern – PlayerManager, BossManager, GameStateManager, HUD. Gebruikt om bepaalde observers (HUD, GameStateManager) op de hoogte te houden van veranderende data zoals aantal munten, levens held, leven boss… en deze dan te gebruiken in de observers.

State pattern – GameStateManager. Gebruikt om bepaalde game states bij te houden (in mijn gevale verschillende schermen)

SOLID voorbeelden:

Single responsibility principe: Gebruikt in klassen zoals: SoundManager, Entities, CollisionHandler, HUD… Deze klassen hebben een eigen verantwoordelijkheid en weten niets over de andere klassen. Kunnen op zichzelf functioneren.

Open/close principe: Entity klasse. Deze is de basis van alle entities en beschrijft hoe een entity functioneert. Kan niet angepast worden, maar wel uitgebreid in subklassen zoals Hero.

Liskov substitution: Entity klasse. Deze klasse is abstract. Bij het gebruiken van de subklassen zoals bepaalde enemies, coins, hero… Maak ik gebruik van de hoofdklasse entity om deze allemaal samen aan te kunnen spreken zoals bv. entity.Update(gameTime);