1
Merge MultiPlayState and SinglePlayState
These two play state have mostly the same functionality, the only difference being a list of Players vs. one Player. The classes were merged by making one class with a list that in the case of a singleplayer game only contains one Player.

Remove Util
The Util class was providing utility for multiple classes all over the project hierarchy. The different functionality was split over the classes it was used by.

2
1.
- The analysis file is located in the `deliverables/inCode/` directory.

2.
- MenuLabel, MenuSelector, MenuComponent are the three classes with the most cumulative flawed classes. Therer are a total of five classes with design flaws.
- The three classes have two design flaws: data clumps, schizophrenic class.

- **Data Clumps** (MenuLabel, MenuSelector, MenuComponent)
  "*The method has a long parameter list, and its signature or a significant fragment thereof is duplicated by other methods. This is a sign that the group of parameters, being passed around collectively to multiple methods in the system, could form a new abstraction, that could be extracted to a new class."* - inCode
- The data clumps are caused by the large constructors of these three classes. For MenuSelector we created the MenuData class which stores 4 integers: x, y, width, and height. This reduces the amount of parameters passed to the constructor.

- **Schizophrenic class** (MenuComponent)
  "The public interface of the class is large and used non-cohesively by client methods i.e., disjoint groups of client classes use disjoint fragments of the class interface in an exclusive fashion." - inCode

- As we browse the cohesive service clusters of this class we we have two clusters of classes accessed from outside: getAbsoluteX and getAbsoluteY. We use these from the superclass MenuComponent. Browsing the local services, namely the public methods that seem to only be used from inside the class gives us the methods: isAcceptingInput, mouseMoved and mousePressed. However: we can not reduce the visibility of the inherited method from InputAdapter. inCode mentiones also seemingly unused services e.g. the abstract process method in MenuComponent. However, this method is used in allmost all menu component related classes. So we obviously need this method. So we do not consider this detected flaw as an error and thus, should not be fixed.

- **Tradition Breaker** (MultiPlayState)
  "The class NOPs or otherwise hides parts of the inherited interface" - inCode

- This design flaw is caused by overriding the `render` and `getScore` method from the `DefaultPlayState` class. This is necessary. The `getScore` method is abstract and needs to be overridden in any case. The `render` method is overridden because the rendering of default attributes is handled in the `DefaultPlayState` method. The render method in the `MultiPlayState` method calls to the DefaultPlayState and then renders `PowerUps`. The `PowerUps` are attributes of `Players` and thus need to be handled in `MultiPlayState`. This is why we chose to ignore this design flaw.

3

1.

Requirements rampage-feature:

Must Haves:

- The game shall have a new mode called ‘Rampage’.
- The ‘Rampage’ mode shall have the player destroy asteroids by hitting them with their plane.
- The ‘Rampage’ mode shall have a time limit of 100 seconds.
- The ‘Rampage’ mode shall its own button in main menu.
- The game will not allow the Player to fire bullets in this game mode
- No powerups will be available
- Changing the difficulty will do nothing for the Rampage mode
- Asteroids will not split into smaller asteroids in this mode

Should Haves:

- The ‘Rampage’ mode shall have a combo multiplier. When the player hits asteroids in quick succession his score will increase by multiplying the multiplier by the base score.

Could Haves:

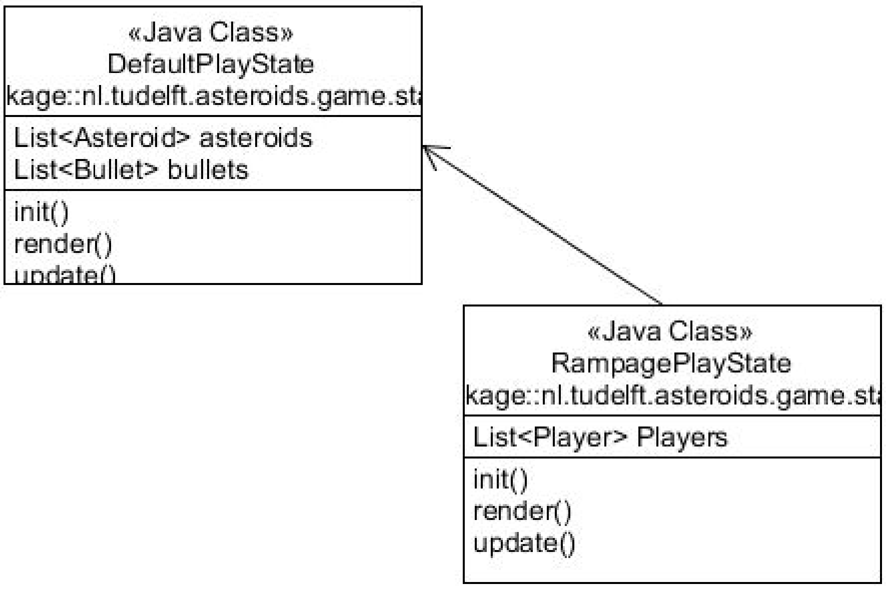
- The ‘Rampage’ mode shall have its own background music.
- The ‘Rampage’ mode shall have special powerups.
- Changing the difficulty of the ‘Rampage’ mode will make it so less powerups spawn

Won’t Haves:

- The ‘Rampage’ mode will have a multiplayer option

2.

UML



CRC

<u>RampagePlayState</u>

DefaultPlayState

-

Process graphics	Render method
Update players, powerups	Update method
Contain players	List element