

**Planet Bound**

Meta 2

Programação Avançada

2019/2020

Ricardo Cardoso 2017017232

Índice

[Opções/Decisões Tomadas na Implementação 3](#_Toc44346015)

[Máquina de Estados 4](#_Toc44346016)

[Classes utilizadas 5](#_Toc44346017)

[Relacionamento das Classes 7](#_Toc44346018)

[Cumprimento das Funcionalidades 8](#_Toc44346019)

## Opções/Decisões Tomadas na Implementação

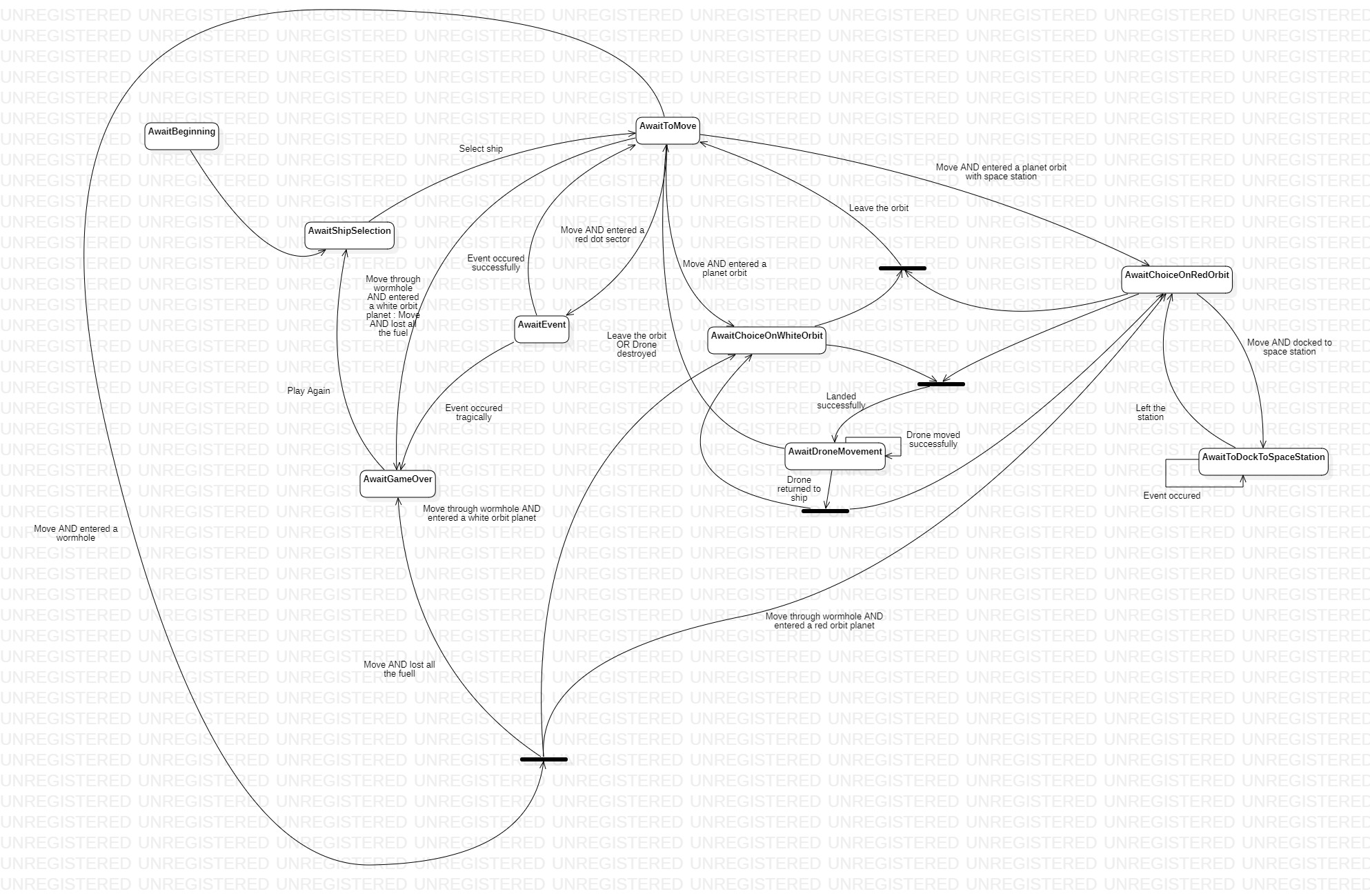
O trabalho prático está organizado de acordo com os padrões lecionados nas aulas, existindo 5 packages: *planetbound, planetbound.logic, planetbound.logic.data, planetbound.states* e *planetbound.ui.text.*

A parte lógica e a interface com o utilizador estão em diferentes packages, permitindo assim o isolamento das mesmas. Foram implementados alguns mecanismos de interação, como menus de texto, para testar o funcionamento da lógica do jogo. Foi também evitado ao máximo qualquer tipo de lógica no package *planetbound.ui.text*.

Relativamente à parte lógica (package *planetbound.logic*), a parte central e funcional do jogo está na classe *GameData*, que é constituída por todos os métodos responsáveis pela execução dos mecanismos do jogo. A lógica de estados está na classe *PlanetBound* e a implementação dos mesmos está na package *planetbound.logic.states*.

A lógica e transição dos estados foi concretizada através de uma máquina de estados, realizada anteriormente.

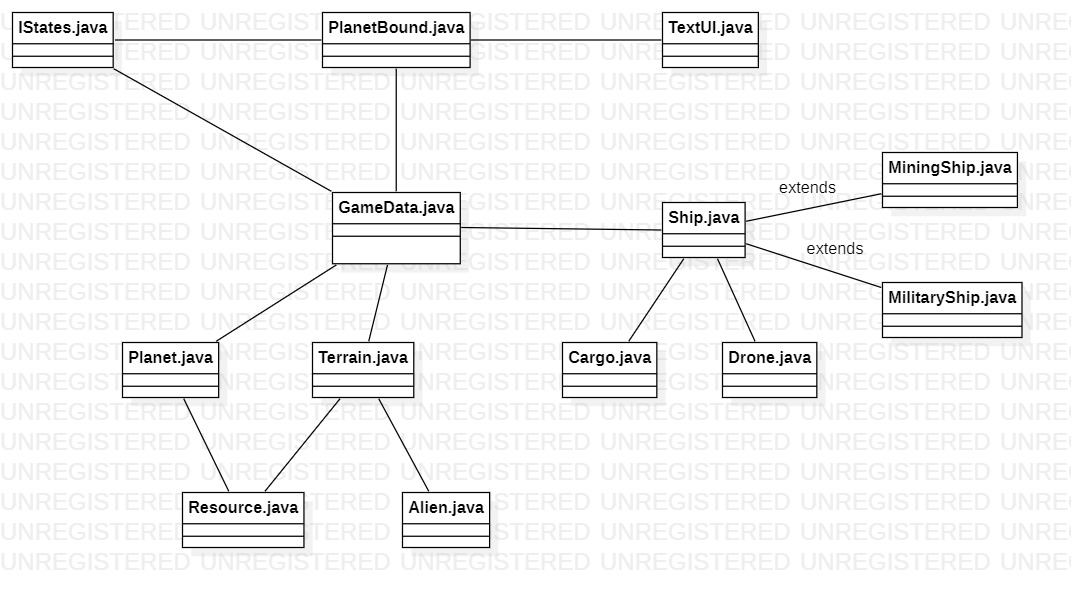
## Máquina de Estados



## Classes utilizadas

* ***planetbound***
  + *Main* – classe onde é dado início ao jogo
* ***planetbound.logic***
  + *PlanetBound* – classe responsável pela transição dos estados.
* ***planetbound.logic.data***
  + *Alien* – classe que representa o alien da superfície do planeta.
  + *Cargo* – classe que representa o armazém de recursos de uma determinada nave.
  + *Constants* – classe constituída por todas as constantes usadas na lógica do jogo.
  + *Drone* – classe que representa o drone de uma determinada nave.
  + *GameData* – classe que contém todos os dados do jogo e métodos que interagem com estes. Responsável pelo funcionamento geral do jogo.
  + *MilitaryShip* – subclasse da classe *Ship* que representa a nave militar.
  + *MiningShip* – subclasse da classe *Ship* que representa a nave exploratória.
  + *Planet* – classe que representa um único planeta.
  + *Resource* – classe que representa um único recurso.
  + *Ship* – superclasse das subclasses *MilitaryShip* e *MiningShip.*
  + *Terrain* – classe que representa a superfície do planeta .
* ***planetbound.logic.states***
  + *AwaitBeginning –* estado que representa a espera do começo do jogo.
  + *AwaitChoiceOnRedOrbit –* estado que representa a espera da indicação do jogador do que fazer quando a nave se encontra na orbita de um planeta com estação espacial.
  + *AwaitChoiceOnWhiteOrbit –* estado que representa a espera da indicação do jogador do que fazer quando a nave se encontra na órbita de um planeta.
  + *AwaitDroneMovemente –* estado que representa a espera do próximo movimento drone.
  + *AwaitEvent –* estado que representa a espera da indicação do jogador do tipo de evento a ocorrer ou se esse evento é gerado aleatoriamente.
  + *AwaitGameOver -* estado que representa a espera da indicação do jogador do que deve fazer após o jogo ter terminado.
  + *AwaitReturnToShip –* estado que representa a espera da indicação do jogador do que fazer quando o drone retorna à nave após uma exploração.
  + *AwaitToDockToSpaceStation –* estado que representa a espera da indicação do jogador do que fazer quando a nave acopla a uma estação espacial.
  + *AwaitToMove –* estado que representa a espera da indicação do jogador se a nave deve prosseguir para a próxima localização.
  + *IStates –* interface constituída por todas as funções implementadas pelos estados.
  + *StateAdapter –* classe que cada um dos estados vai estender de forma a só ter de fazer *override* das funções que necessita.
* ***planetbound.ui.text***
  + *TextUI –* classe que representa a interface de texto do jogo.

## Relacionamento das Classes



## Cumprimento das Funcionalidades

|  |  |
| --- | --- |
| TAREFA | CUMPRIDO |
| 1. Devem ser seguidos os padrões apresentados nas aulas. | Sim |
| 2. Deve ser utilizada, de forma adequada, uma máquina de estados para concretizar a lógica do jogo. | Sim |
| 3. Ao terminar um jogo, a máquina de estados deve permitir ambas as hipóteses de jogar de novo ou de terminar a aplicação | Sim |
| 4. Na fase Alien Attacks, a sequência de ações correspondentes ao ataque do alienígena e do drone, que não dependem de qualquer decisão por parte do jogador, devem ser automaticamente desencadeadas sem qualquer intervenção do jogador. Os momentos do jogo em que o jogador tem que tomar decisões devem corresponder a estados. Podem existir estados em que o jogador é informado acerca do ponto da situação e apenas decide quando prosseguir. A existência de estados nestas situações depende apenas da dinâmica que pretendemos dar à interação com o utilizador; | Sim |
| 5. A aplicação deve apresentar a informação necessária ao acompanhamento e verificação do bom funcionamento do jogo (dados relevantes que caracterizam as diversas cartas usadas no jogo, o turno e a fase atual, as situações de vitória e derrota, etc.). Relativamente ao assunto da apresentação de informação (interface com o utilizador), chama-se a atenção para o fato do ecrã não ter as mesmas limitações de um tabuleiro físico. Significa isto que a interface pode apresentar a mesma informação de forma mais compacta ou inteligente. Por exemplo, em vez de uma série de quadrados para representar combustível ou ocupação do porão, um indicador numérico ou um ponteiro como o do combustível nos automóveis cumprirão o mesmo objetivo de uma forma mais interessante e visualmente mais agradável; | Sim |
| 6. O modelo do jogo (máquina de estados) deve, através de um log (“registo”) interno (não é permitida qualquer tipo de interação com o utilizador), gerar e disponibilizar (note que “disponibilizar” não significa “mostrar”, mas sim “tornar acessível”) informação detalhada sobre o estado interno e o resultado das várias ações. Por exemplo, deve ser possível a interface do utilizador obter os detalhes do que ocorreu automaticamente. | Sim |
| 7. Em ambas as fases do trabalho, deve ser utilizado o padrão de separação entre Modelo e Vista estudado nas aulas. | Sim |