Meeting

User Interface

Use case

- Add players
- Launch game
- Move ships
- Access ports
- Get chance cards/treasure
- Battle other players
- Zoom
- End turn
- Trade
- Access islands

Typical users

- 8+
- People who can use a computer
- …?

Error conditions

- Invalid name
- Tries to move and turn ship
- Drawing multiple crew card
- Having more the allowed number of crew cards

Main menu

Sends player to name select menu

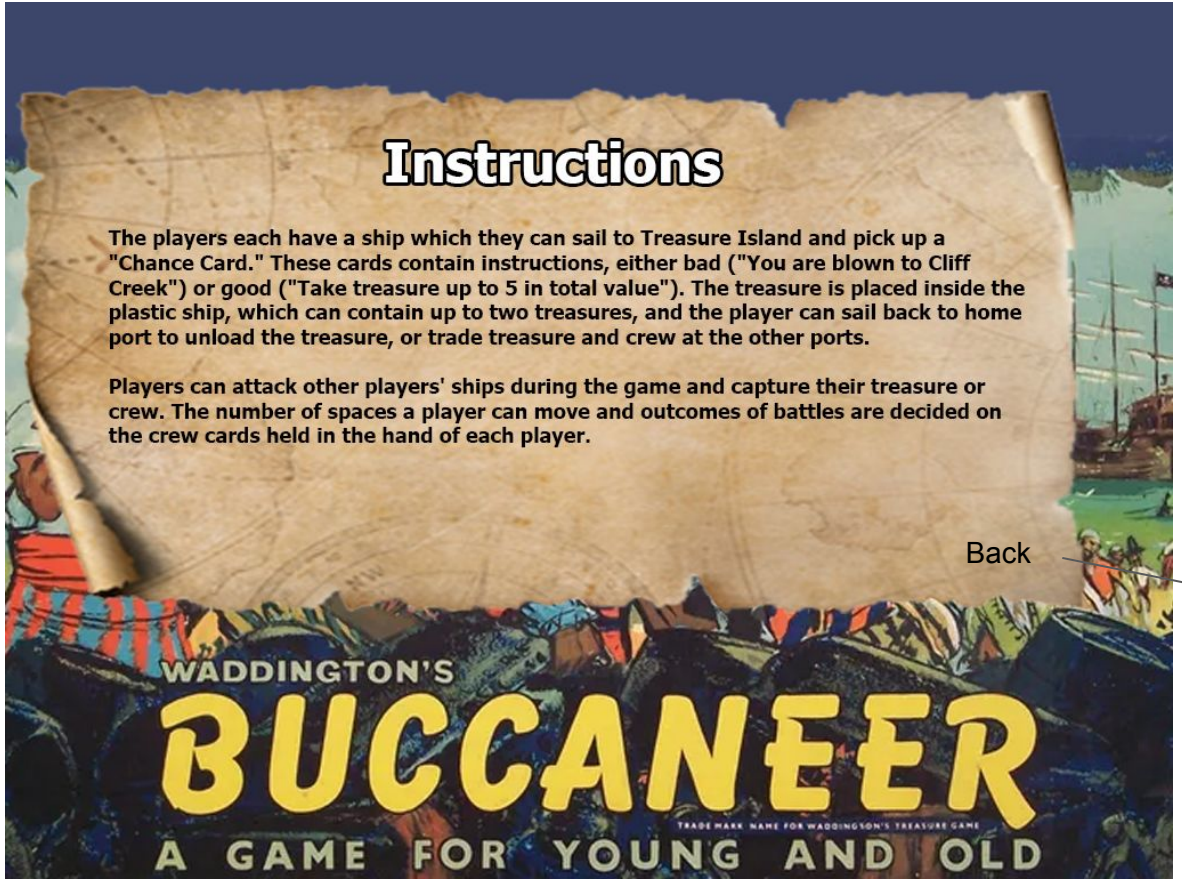
Opens Instructions

Player name select

Players type their names in

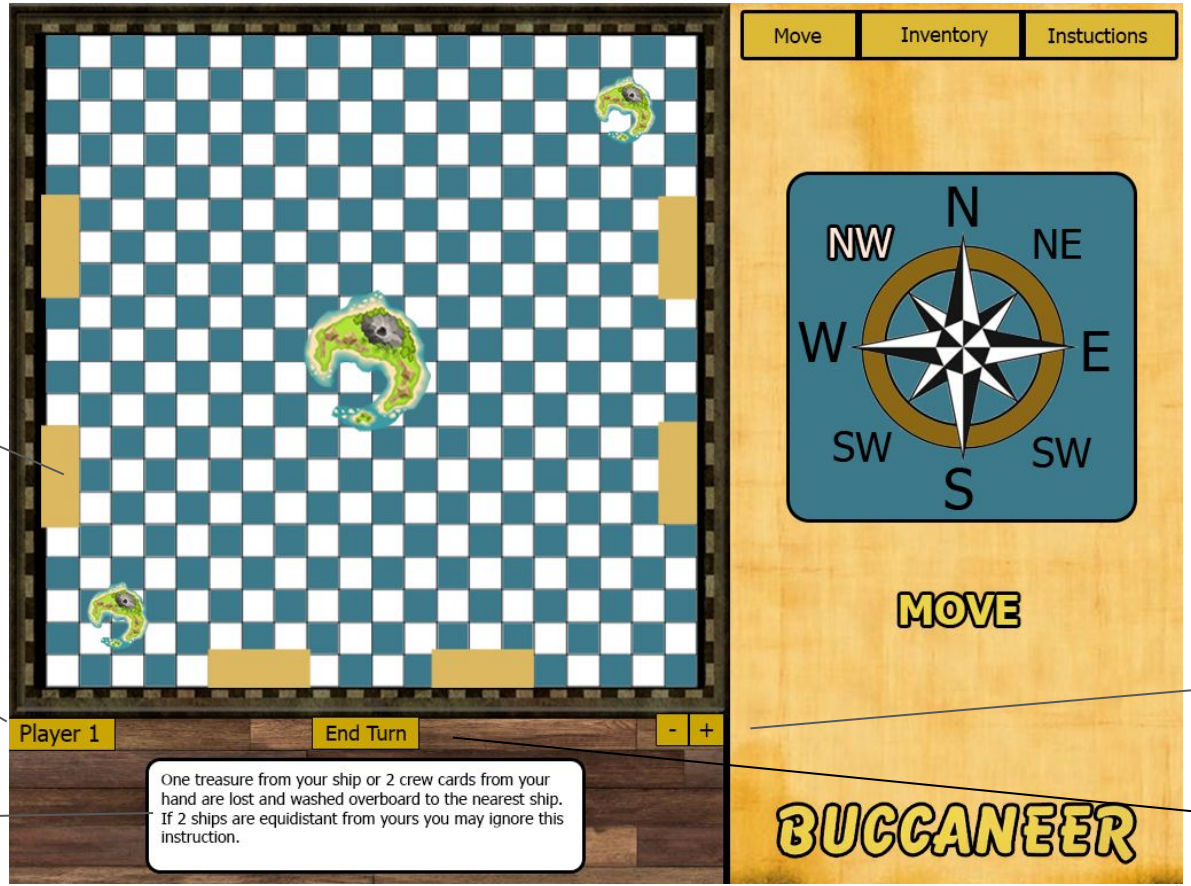
Launches game

Instruction Menu



Takes you back to the main menu

Game board



Menu

Dock

Control movement

Current player

Zoom

End turn button

Card text

Move | Inventory | Instuctions

Cards

Treasure

Diamond - 4

Ruby - 1

Barrel - 2

The player's crew cards

Player's treasure

BUCCANEER

Move | Inventory | Instuctions

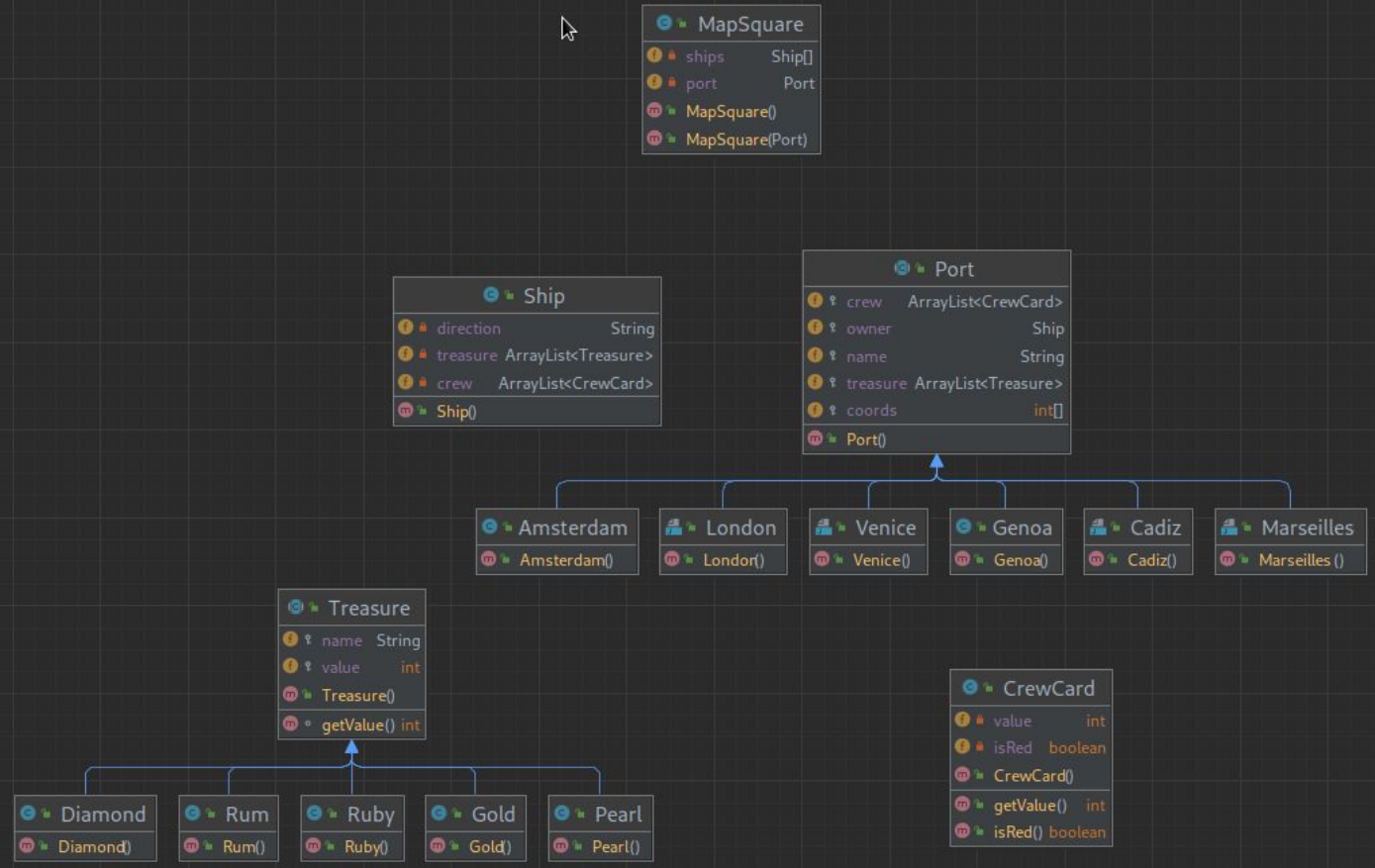
The players each have a ship which they can sail to Treasure Island and pick up a "Chance Card." These cards contain instructions, either bad ("You are blown to Cliff Creek") or good ("Take treasure up to 5 in total value"). The treasure is placed inside the plastic ship, which can contain up to two treasures, and the player can sail back to home port to unload the treasure, or trade treasure and crew at the other ports.

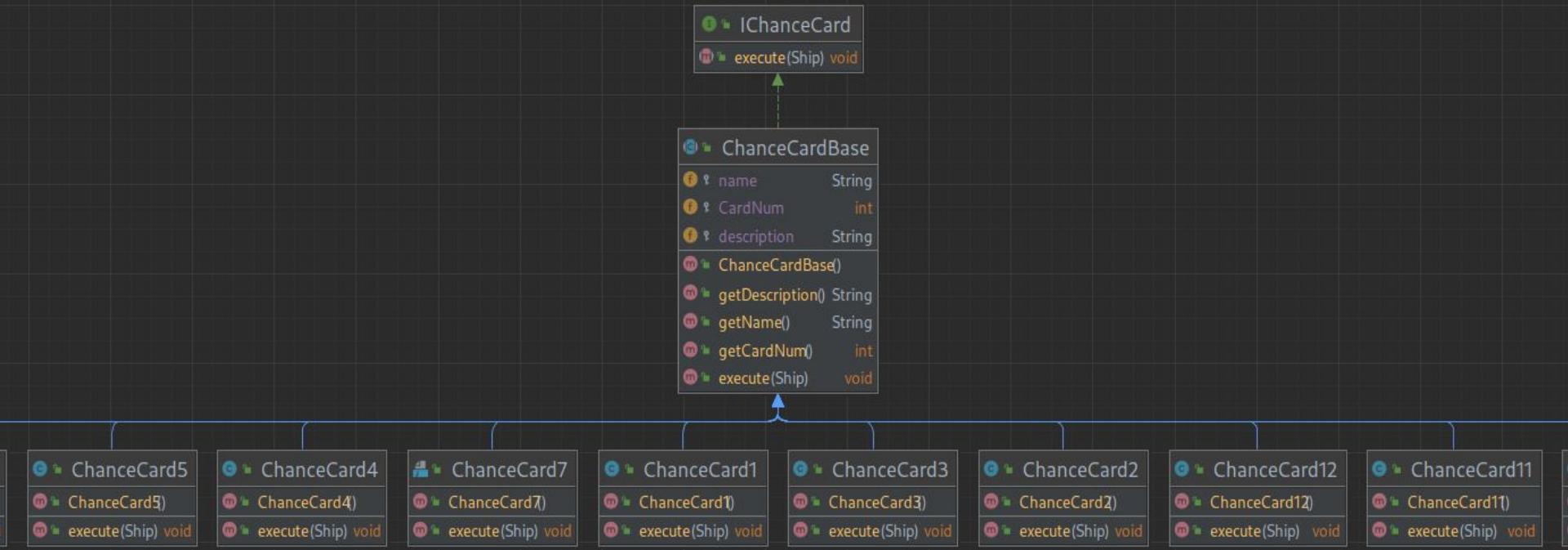
Players can attack other players' ships during the game and capture their treasure or crew. The number of spaces a player can move and outcomes of battles are decided on the crew cards held in the hand of each player.

Instructions

BUCCANEER

Objects and Interactions


MapSquare
ships Ship[]
port Port
MapSquare()
MapSquare(Port)
Ship
direction String
treasure ArrayList<Treasure>
crew ArrayList<CrewCard>
Ship()
Port
crew ArrayList<CrewCard>
owner Ship
name String
treasure ArrayList<Treasure>
coords int[]
Port()
Amsterdam
Amsterdam()
London
London()
Venice
Venice()
Genoa
Genoa()
Cadiz
Cadiz()
Marseilles
Marseilles ()
Treasure
name String
value int
Treasure()
getValue() int
Diamond
Diamond()
Rum
Rum()
Ruby
Ruby()
Gold
Gold()
Pearl
Pearl()
CrewCard
value int
isRed boolean
CrewCard()
getValue() int
isRed() boolean


IChanceCard
execute(Ship) void
ChanceCardBase
name String
CardNum int
description String
ChanceCardBase()
getDescription() String
getName() String
getCardNum() int
execute(Ship) void
ChanceCard5
ChanceCard5()
execute(Ship) void
ChanceCard4
ChanceCard4()
execute(Ship) void
ChanceCard7
ChanceCard7()
execute(Ship) void
ChanceCard1
ChanceCard1()
execute(Ship) void
ChanceCard3
ChanceCard3()
execute(Ship) void
ChanceCard2
ChanceCard2()
execute(Ship) void
ChanceCard12
ChanceCard12()
execute(Ship) void
ChanceCard11
ChanceCard11()
execute(Ship) void

Test Spec draft

Code example

```java
/**
 * Making sure at the start off the game empty ports are not assigned to any players
 * **/
@Test
public void testEmptyLondon(){
    Ship testShip = new Ship();
    //testShip.playerName = "";
    Port testPort = new London();
    London.owner = testShip;
    assertTrue(testShip.playerName.isEmpty());

}
```

```java
@Test
public void testEmptyGenoa(){
    Ship testShip = new Ship();
    Port testPort = new Genoa();
    Genoa.owner = testShip;
    assertTrue(testShip.playerName.isEmpty());

}
```

```java
@Test
public void testEmptyMarseilles(){
    Ship testShip = new Ship();
    Port testPort = new Marseilles();
    Marseilles.owner = testShip;
    assertTrue(testShip.playerName.isEmpty());

}
```

```java
@Test
public void testEmptyCadiz(){
    Ship testShip = new Ship();
    Port testPort = new Cadiz();
    Cadiz.owner = testShip;
    assertTrue(testShip.playerName.isEmpty());

}
```