**THE PLEASE DON’T SINK ME SIMULATOR**

[*Game rules will follow the rules of the original game from Hasbro*](http://www.hasbro.com/common/instruct/battleship.pdf)*.*

*Written by*

*René Flores, Omar Sanchez, Will Awad*

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\

DEFINITIONS OF OBJECTS TO BE USED

\\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**//Definition of Commander Objects**

{

int sp //ship points determine the amount of “health”

}

**//Definition of Ship Objects**

{

string name //determines the name of the ship

int size //determines how many spaces it takes up on the map

string icon //determines how the ship will look in the map

}

**//Definition of Map Objects**

{

array[array[String]] //will store strings within a 2D array

formatCoords(string coords, int half) //takes coords and int to determine that part of the coord to return

{

if int half is 1

take first half of string, parse it as int

return int

else if int half is 2

take second of string, parse it as int

return int

}

printMap() //shows the map on console

{

print top border

for every column in map

print left border

for every string in row of map

print space

print icon

print space

print border

print right border

print bottom border

}

}

**//Definition of shipMap objects inherited from Map**

{

checkPosition() //checks to see if current coords are adjacent to previous coords

{

prompt for coords and save as string coords

call formatCode and give coords and 1 as parameter

save it as int Y

call formatCode and give coords and 2 as parameter

save it as int X

if int X - 1 and Y + 0 is same ship object

return true

else if int X + 1 and Y + 0 is same ship object

return true

else if int X + 0 and Y + 1 is same ship object

return true

else if int X + 0 and Y - 1 is same ship obejct

return true

else

return false

}

placePlayerCoords() //validates, formats, places string in map

{

while coordinate validity is false

prompt for coords and save as string coords

call parent formatCode and give coords and 1 as parameter

save it as int Y

call parent formatCode and give coords and 2 as parameter

save it as int X

if the X or Y coordinate is out of bounds of the map

give "out of bounds" error message

ask them to place again

else if it's in bounds but coordinate is not "water" icon

give "space is taken" error message

ask them to place again

else

put string representation in coordinates of map

coordinate validity is true

}

placeEnemyCoords() //enemy AI chooses random coords

{

while coordinate validity is false

generate random number within bounds and pass to Y

generate random number within bounds and pass to X

if the coordinate is not "water" icon

go back to the start of loop

else

put string representation in coordinates of map

coordinate validity is true

}

}

**//Definition of choiceMap inherited from Map**

{

playerAttack()

{

while choice validity is false

prompt user to input coordinates as string

call parent formatCode and give coords and 1 as parameter

save it as int Y

call parent formatCode and give coords and 2 as parameter

save it as int X

if the X and Y coordinate equal “hit” or “miss” mark on playerChoiceMap

return to the top of the loop

else if X and Y coordinate equal “water” mark on enemyShipMap

place “miss” mark on enemyShipMap of that coordinate

place “miss” mark on playerChoiceMap of that coordinate

choice validity is set to true

else if X and Y coordinate equal any ship icon on enemyShipMap

place “hit” mark on enemyShipMap of that coordinate

place “hit” mark on playerChoiceMap of that coordinate

subtract 1 SP from enemy

choice validity is set to true

}

enemyAttack()

{

while choice validity is false

generate random number within bounds and save as int X

generate random number within bounds and save as int Y

if the X and Y coordinate equal “hit” or “miss” mark on enemyChoiceMap

return to the top of the loop

else if X and Y coordinate equal “water” mark on playerShipMap

place “miss” mark on playerShipMap of that coordinate

place “miss” mark on enemyChoiceMap of that coordinate

choice validity is set to true

else if X and Y coordinate equal any ship icon on playerShipMap

place “hit” mark on playerShipMap of that coordinate

place “hit” mark on enemyChoiceMap of that coordinate

subtract 1 SP from enemy

choice validity is set to true

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\

SEQUENCE OF THE GAME

\\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**//Introduction Phase**

print introduction screen

print rules screen

print key screen

**//Commander Creation Phase**

create Commander player with sp = 17 //makes player as commander, sp equal to all ship size

create Commander enemy with sp = 17 //makes enemy as commander, sp equal to all ship sizes

**//Map Creation Phase**

create Map playerChoiceMap //map for player to guess on

create Map enemyChoiceMap //map for enemy to guess on

create Map playerShipMap //map for player's ships

create Map enemyShipMap //map for enemy's ships

**//Ship Creation Phase**

createFleet()

{

//creates ship objects and returns them in an array of ship objects.

create ship with name = Destroyer, size = 2, icon = D

create ship with name = Submarine, size = 3, icon = S

create ship with name = Cruiser, size = 3, icon = Z

create ship with name = Battleship, size = 4, icon = B

create ship with name = Carrier, size = 5, icon = R

create array of objects

append Destroyer to array

append Submarine to array

append Cruiser to array

append Battleship to array

append Carrier to array

return array of ships

}

**//Preparation Phase**

playerPrepPhase(Array of Ship objects, Map playerShipMap)

{

//initiates player prep phase to choose where to place their ships

//also keeps tracks if it's the first time placing a ship to then make sure the next choices are next to each other

for every ship object in the array

print the player's shipMap

for loop, variable equals size of ship, until variable decrements to 0

if variable is still equal to size of ship

call Map function placePlayerCoords

else

while next choice validity is false

call Map function checkPosition

and save it to choice validity

call Map function placePlayerCoords

}

enemyPrepPhase(Array of Ship objects, Map enemyShipMap)

{

//initiates enemy prep phase

//also keeps track of number of times cycling through

for every ship object in the array

for loop, variable equals size of ship, until variable decrements to 0

if variable is still equal to size of ship

call Map function placeEnemyCoords

else

while next choice validity is false

call Map function checkPosition

and save it to choice validity

call Map function placeEnemyCoords

}

**//Battle Phase**

while player’s SP is not 0 or enemy’s SP is not 0

if playerTurn is true

call choiceMap function *playerAttack*

set playerTurn to false

else if playerTurn is false

call choiceMap function *enemyAttack*

set playerTurn to true

**//Ending Phase**

if player SP equals 0

print lose screen

else if enemy SP equals 0

print victory screen

print credits