BATTLESHIP CLASS

```
import java.util.Scanner;

public class Battleship
{
    //Global variables
    static Scanner input = new Scanner(System.in);
    static Player player = new Player();
    static Player computer = new Player();
    static Ship[] playerShips = player.getShip();
    static Randomizer rand = new Randomizer();

    public static void main(String[] args) {

        int playerScore = 0;
        int computerScore = 0;

        player.printMyShips();
        System.out.println("Which row would you like to place your ship? Please choose 
between 0-9.");
        int row = input.nextInt();
        System.out.println("Which column would you like to place your ship? Please choose 
between 0-9.");
        int col = input.nextInt();
        System.out.println("Which direction would you place your ship? Please choose 0 
for horizontally or 1 for vertically?");
        int direction = input.nextInt();
        player.chooseShipLocation(playerShips[0], row, col, direction);

        player.printMyShips();
        System.out.println("Which row would you like to place your ship? Please choose 
between 0-9.");
        row = input.nextInt();
        System.out.println("Which column would you like to place your ship? Please choose 
between 0-9.");
        col = input.nextInt();
        System.out.println("Which direction would you place your ship? Please choose 0 
for horizontally or 1 for vertically");
        direction = input.nextInt();
        player.chooseShipLocation(playerShips[1], row, col, direction);

        player.printMyShips();
        System.out.println("Which row would you like to place your ship? Please choose 
between 0-9.");
        row = input.nextInt();
        System.out.println("Which column would you like to place your ship? Please choose 
between 0-9.");
        col = input.nextInt();
        System.out.println("Which direction would you place your ship? Please choose 0 
for horizontally or 1 for vertically");
        direction = input.nextInt();
        player.chooseShipLocation(playerShips[2], row, col, direction);

        player.printMyShips();
        System.out.println("Which row would you like to place your ship? Please choose 
between 0-9.");
        row = input.nextInt();
        System.out.println("Which column would you like to place your ship? Please choose 
between 0-9.");
        col = input.nextInt();
        System.out.println("Which direction would you place your ship? Please choose 0 
for horizontally or 1 for vertically");
        direction = input.nextInt();
        player.chooseShipLocation(playerShips[3], row, col, direction);

        player.printMyShips();
        System.out.println("Which row would you like to place your ship? Please choose 
between 0-9.");
        row = input.nextInt();
        System.out.println("Which column would you like to place your ship? Please choose 
between 0-9.");
        col = input.nextInt();
        System.out.println("Which direction would you place your ship? Please choose 0 
for horizontally or 1 for vertically");
        direction = input.nextInt();
        player.chooseShipLocation(playerShips[4], row, col, direction);
        System.out.println("Player:");
        player.printMyShips();

        computer.chooseShipLocation(playerShips[0], 9, 1, 0);
        computer.chooseShipLocation(playerShips[1], 1, 2, 0);
        computer.chooseShipLocation(playerShips[2], 2, 8, 1);
        computer.chooseShipLocation(playerShips[3], 5, 7, 1);
        computer.chooseShipLocation(playerShips[4], 4, 1, 0);
        System.out.println("Computer:");
        computer.printMyShips();

        while(playerScore != 17 && computerScore != 17)
        {
            System.out.println("What row (Numbers 0-9) do you think the opponent has 
ships at?");
            row = input.nextInt();
            System.out.println("What column (Numbers 0-9) do you think the opponent has 
ships at??");
            col = input.nextInt();
            if (computer.grid1.hasShip(row, col))
            {
                player.grid2.markHit(row, col);
                computer.grid1.markHit(row, col);
                playerScore++;
            }
            else
            {
                player.grid2.markMiss(row, col);
                computer.grid1.markMiss(row, col);
            }
            player.printMyGuesses();

            row = rand.nextInt(0, 9);
            col = rand.nextInt(0, 9);
            if (player.grid1.hasShip(row, col))
            {
                computer.grid2.markHit(row, col);
                player.grid1.markHit(row, col);
                computerScore++;
            }
            else
            {
                computer.grid2.markMiss(row, col);
                player.grid1.markMiss(row, col);
            }
            player.printOpponentGuesses();
        }
        if(playerScore == 17)
        {
            System.out.println("You Win!");
        }
        else if(computerScore == 17)
        {
            System.out.println("You lose");
        }
    }
}
```

LOCATION CLASS

```
public class Location
{
    public static final int UNGUESSED = 0;
    public static final int HIT = 1;
    public static final int MISSED = 2;

    private boolean ship = false;
    private int status = UNGUESSED;

    // Location constructor.
    public Location()
    {

    }

    // Checks to see status of location and if it was a hit or not
    public boolean checkHit()
    {
        if(status == HIT)
        {
```

```
        return(true);
    }
    else
    {
        return(false);
    }
}

// Checks to see status of location and if it was a miss or not
public boolean checkMiss()
{
    if(status == MISSED)
    {
        return(true);
    }
    else
    {
        return(false);
    }
}

// Checks to see if this lcation was UNGUESSED
public boolean isUnguessed()
{
    if(status == UNGUESSED)
    {
        return(true);
    }
    else
    {
        return(false);
    }
}

// If it was a hit, it marks the location
public void markHit()
{
    status = HIT;
}

// If it was not a hit, it marks the location a miss
public void markMiss()
{
    status = MISSED;
}


public boolean hasShip()
{
    return(ship);
}
```

```
    public void setShip(boolean val)
    {
        ship = val;
    }

    public void setStatus(int stat)
    {
        status = stat;
    }

    public int getStatus()
    {
        return status;
    }
}
```

GRID CLASS

```
public class Grid
{
    public static final int UNSET = -1;
    public static final int HORIZONTAL = 0;
    public static final int VERTICAL = 1;

    public static final int UNGUESSED = 0;
    public static final int HIT = 1;
    public static final int MISSED = 2;

    public static final int NUM_ROWS = 10;
    public static final int NUM_COLS = 10;
    public static final String verticalList[] = {"0", "1", "2", "3", "4", "5", "6", "7",
"8", "9"};

    private Location[][] grid = new Location[NUM_ROWS][NUM_COLS];

    // Create a new Grid. Initialize each Location in the grid
    // to be a new Location object.
    public Grid()
    {
        for(int r = 0; r < NUM_ROWS; r++)
        {
            for(int c = 0; c < NUM_COLS; c++)
            {
                grid[r][c] = new Location();
            }
        }
    }

    // Mark a hit in this location by calling the markHit method on the Location object.
    public void markHit(int row, int col)
    {
        grid[row][col].markHit();
```

```java
}

// Mark a miss on this location.
public void markMiss(int row, int col)
{
    grid[row][col].markMiss();
}

// Set the status of this location object.
public void setStatus(int row, int col, int status)
{
    grid[row][col].setStatus(status);
}

//Gets status of the grid
public int getStatus(int row, int col)
{
    return grid[row][col].getStatus();
}

public boolean alreadyGuessed(int row, int col)
{
    return !grid[row][col].isUnguessed();
}

public void setShip(int row, int col, boolean val)
{
    grid[row][col].setShip(val);
}

public boolean hasShip(int row, int col)
{
    return grid[row][col].hasShip();
}

public Location get(int row, int col)
{
    return grid[row][col];
}

public int numRows()
{
    return NUM_ROWS;
}

public int numCols()
{
    return NUM_COLS;
}

public void addShip(Ship s)
```

```java
{
    int row = s.getRow();
    int col = s.getCol();
    int length = s.getLength();
    int direction = s.getDirection();

    if(direction == HORIZONTAL)
    {
        for(int c = col; c < col + length; c++)
        {
            setShip(row,c,true);
        }
    }
    else
    {
        for(int r = row; r < row + length; r++)
        {
            setShip(r,col,true);
        }
    }
}

// Function prints the grid and any updates in statuses
public void printStatus()
{
    System.out.println("  0 1 2 3 4 5 6 7 8 9");
    for(int r = 0; r < NUM_ROWS; r++)
    {
        System.out.print(verticalList[r] + " ");
        for(int c = 0; c < NUM_COLS; c++)
        {
            if(!alreadyGuessed(r,c))
            {
                System.out.print("- ");
            }
            else if(getStatus(r,c) == HIT)
            {
                System.out.print("! ");
            }
            else if(getStatus(r,c) == MISSED)
            {
                System.out.print("O ");
            }
        }
        System.out.println();
    }
}

// Function prints the grid and any updates in statuses
public void printShips()
{
    System.out.println("  0 1 2 3 4 5 6 7 8 9");
    for(int r = 0; r < NUM_ROWS; r++)
```

```
            {
                System.out.print(verticalList[r] + " ");
                for(int c = 0; c < NUM_COLS; c++)
                {
                    if(hasShip(r,c))
                    {
                        System.out.print("X ");
                    }
                    else
                    {
                        System.out.print("- ");
                    }
                }
                System.out.println();
            }
        }
    }
```

PLAYER CLASS

```
public class Player
{
    private static final int[] SHIP_LENGTHS = { 2, 3, 3, 4, 5 };
    private static final int NUM_SHIPS = 5;

    public Grid grid1;
    public Grid grid2;
    private final Ship[] myShips;

    public Player()
    {
        grid1 = new Grid();
        grid2 = new Grid();
        myShips = new Ship[NUM_SHIPS];
        for(int i = 0; i < NUM_SHIPS; i++)
        {
            Ship temp = new Ship(SHIP_LENGTHS[i]);
            myShips[i] = temp;
        }
    }

    public Ship[] getShip()
    {
        return myShips.clone();
    }

    public void chooseShipLocation(Ship s, int row, int col, int direction)
    {
        s.setLocation(row, col);
        s.setDirection(direction);
        grid1.addShip(s);
    }
    //Prints player ships on player grid
    public void printMyShips()
    {
        grid1.printShips();
    }

    //Print opponents guesses on player grid
    public void printOpponentGuesses()
    {
        grid1.printStatus();
    }

    //Print player guesses on opponent grid
    public void printMyGuesses()
    {
        grid2.printStatus();
    }

    /*public boolean recordMyGuess(int row, int col)
    {
        boolean guessMiss = grid2.hasShip(row, col);
        if(guessMiss)
        {
            grid2.markHit(row, col);
        }
        else
        {
            grid2.markMiss(row, col);
        }
        return guessMiss;
    }
*/
    //Records where the opponent has guessed
   /* public boolean recordOpponentGuess(int row, int col)
    {
        boolean guessMiss = grid1.hasShip(row, col);
        if(guessMiss)
        {
            grid1.markHit(row, col);
        }
        else
        {
            grid1.markMiss(row, col);
        }
        return guessMiss;*/
    }
```

RANDOMIZER CLASS (separated player class, computer player is randomizer)

```
//This will help simulate the computer playing the second role. Creating a randomizer
allows for no bias and "computer" player
import java.util.*;

public class Randomizer{
    public static Random theInstance = null;
    public Randomizer(){
    }

    public static Random getInstance(){
        if(theInstance == null){
            theInstance = new Random();
        }
        return theInstance;
    }

    public static boolean nextBoolean(){
        return Randomizer.getInstance().nextBoolean();
    }

    public static boolean nextBoolean(double probability){
        return Randomizer.nextDouble() < probability;
    }


    public static int nextInt(){
        return Randomizer.getInstance().nextInt();
    }

    public static int nextInt(int n){
        return Randomizer.getInstance().nextInt(n);
    }

    public static int nextInt(int min, int max){
        return min + Randomizer.nextInt(max - min + 1);
    }

    public static double nextDouble(){
        return Randomizer.getInstance().nextDouble();
    }

    public static double nextDouble(double min, double max){
        return min + (max - min) * Randomizer.nextDouble();
    }


}
```

SHIP CLASS

```
//This class creates the characteristics and details of the ship.
public class Ship
{
    public static final int UNSET = -1;
    public static final int HORIZONTAL = 0;
    public static final int VERTICAL = 1;

    private int row = UNSET;
    private int col = UNSET;
    private int length = UNSET;
    private int direction = UNSET;

    // Creates ship and length of ship
    public Ship(int shipLength)
    {
        length = shipLength;
    }

    public boolean isLocationSet()
    {
        if(row == UNSET || col == UNSET)
        {
            return(false);
        }
        else
        {
            return(true);
        }
    }

    public boolean isDirectionSet()
    {
        if(direction == UNSET)
        {
            return(false);
        }
        else
        {
            return(true);
        }
    }

    // Creates ship direction
    public void setLocation(int r, int c)
    {
        row = r;
        col = c;
    }

    // Creates ship direction
    public void setDirection(int d)
    {
```

```
    direction = d;
}

// Gets the user row value
public int getRow()
{
    return(row);
}

// Gets the user column value
public int getCol()
{
    return(col);
}

// Gets the length of ship
public int getLength()
{
    return(length);
}

// Gets the direction of ship
public int getDirection()
{
    return(direction);
}

// Function that translates user input (0 or 1) to direction (horizontal / vertical)
private String directionToString()
{
    if(direction == HORIZONTAL)
    {
        return("horizontal");
    }
    else if(direction == VERTICAL)
    {
        return("vertical");
    }
    else
    {
        return("unset direction");
    }
}

private String locationToString()
{
    if(isLocationSet())
    {
        return("(" + row + ", " + col + ")");
    }
    else
    {
            return("(unset location)");
        }
    }

    // Ship length to string
    public String toString()
    {
        return(directionToString() + " ship of length " + length + " at " +
locationToString());
    }
}
```