```
using System;
using System.Collections.Generic;

namespace Battleships
{
    class Location
    {
        private int x;
        private int y;

        private Dictionary<char, int> guessRow = new Dictionary<char, int>()
        {
            {'A', 0},
            {'B', 1},
            {'C', 2},
            {'D', 3},
            {'E', 4},
            {'F', 5},
            {'G', 6},
            {'H', 7},
            {'I', 8},
            {'J', 9}
        };

        public Location(int x, int y)
        {
            this.x = x;
            this.y = y;
        }

        public Location(String guessLocation)
        {
            try
            {
                this.y = guessRow[guessLocation[0]];
                this.x = (int)char.GetNumericValue(guessLocation[1]);
            }
            catch(System.Exception)
            {
                throw;
            }
        }

        public int getX()
        {
            return x;
        }

        public void setX(int x)
        {
            this.x = x;
        }

        public int getY()
        {
            return y;
        }

        public void setY(int y)
        {
            this.y = y;
        }
    }

    abstract class Board
    {
        private Object[,] board = new Object[10, 10];

        public Object getCell(Location location)
        {
            return board[location.getX(), location.getY()];
        }

        public void setCell(Location location, Object cellContents)
        {
            board[location.getX(), location.getY()] = cellContents;
        }

        protected void initialise(Object initial)
        {
            for (int x = 0; x < 10; x++)
            {
                for (int y = 0; y < 10; y++)
                {
                    setCell(new Location(x, y), initial);
                }
            }
        }

        public void showBoard()
        {
            Console.WriteLine("+-------------------------------------------+");
            Console.WriteLine("|   | A | B | C | D | E | F | G | H | I | J |");
            Console.WriteLine("+-------------------------------------------+");

            for (int x = 0; x < 10; x++)
            {
                Console.Write("| " + x + " | ");

                for (int y = 0; y < 10; y++)
                {
                    Location temp = new Location(x, y);
                    Console.Write(displayCell(temp) + " | ");
                }

                Console.WriteLine("\n +-------------------------------------------+");
            }
        }
```

```csharp

        protected abstract char displayCell(Location location);
    }

    class Ship
    {
        protected int length;
        protected char shipSymbol;

        public int getLength()
        {
            return length;
        }

        public char getShipSymbol()
        {
            return shipSymbol;
        }
    }

    class CarrierShip : Ship
    {
        public CarrierShip()
        {
            length = 5;
            shipSymbol = 'A';
        }
    }

    class BattleShip : Ship
    {
        public BattleShip()
        {
            length = 4;
            shipSymbol = 'B';
        }
    }

    class CruiserShip : Ship
    {
        public CruiserShip()
        {
            length = 3;
            shipSymbol = 'C';
        }
    }

    class SubmarineShip : Ship
    {
        public SubmarineShip()
        {
            length = 3;
            shipSymbol = 'S';
        }
    }

    class DestroyerShip : Ship
    {
        public DestroyerShip()
        {
            length = 2;
            shipSymbol = 'D';
        }
    }

    class ShipBoard : Board
    {
        private int totalShips;

        public ShipBoard()
        {
            initialise(null);

            Random random = new Random();

            totalShips = 0;

            List<Ship> ships = new List<Ship>()
            {
                new DestroyerShip(),
                new SubmarineShip(),
                new CruiserShip(),
                new BattleShip(),
                new CarrierShip()
            };

            int locationX;
            int locationY;
            int direction;
            List<Location> shipLocations;

            foreach (Ship ship in ships)
            {
                bool shipPlaced = false;

                while (!shipPlaced)
                {
                    locationX = random.Next(0, 9);
                    locationY = random.Next(0, 9);
                    direction = random.Next(0, 3);
                    shipLocations = new List<Location>();

                    switch (direction)
                    {
                        case 1:
                            if (locationX - ship.getLength() > 0)
                            {
                                for (int x = locationX; x > locationX - 
                        ship.getLength(); x --)
                                {
                                    addShip(new Location(x, locationY), 
                        shipLocations);
```

```
                                }

                                shipPlaced = placeShips(shipLocations, ship);
                            }
                            break;
                        case 2:
                            if (locationY + ship.getLength() < 10)
                            {
                                for (int y = locationY; y < locationY + ship.getLength(); y++)
                                {
                                    addShip(new Location(locationX, y), shipLocations);
                                }

                                shipPlaced = placeShips(shipLocations, ship);
                            }
                            break;
                        case 3:
                            if (locationX + ship.getLength() < 10)
                            {
                                for (int x = locationX; x < locationX + ship.getLength(); x++)
                                {
                                    addShip(new Location(x, locationY), shipLocations);
                                }

                                shipPlaced = placeShips(shipLocations, ship);
                            }
                            break;
                        case 4:
                            if (locationY - ship.getLength() > 0)
                            {
                                for (int y = locationY; y > locationY - ship.getLength(); y--)
                                {
                                    addShip(new Location(locationX, y), shipLocations);
                                }

                                shipPlaced = placeShips(shipLocations, ship);
                            }
                            break;
                    }
                }
            }
        }

        private bool placeShips(List<Location> locations, Ship ship)
        {
            if (locations.Count == ship.getLength())
            {
                foreach (Location location in locations)
                {
                    setCell(location, ship);
                    totalShips++;
                }
                return true;
            }

            return false;
        }

        private void addShip(Location location, List<Location> shipLocations)
        {
            if (getCell(location) == null)
            {
                shipLocations.Add(location);
            }
        }

        public int getTotalShips()
        {
            return totalShips;
        }

        public Boolean hasHit(Location guessLocation)
        {
            return (getCell(guessLocation) != null);
        }

        protected override char displayCell(Location location)
        {
            Object ship = getCell(location);
            return (ship == null ? '-' : ((Ship)ship).getShipSymbol());
        }
    }

    class Guess
    {
        private char guess;

        public Guess(char guess)
        {
            this.guess = guess;
        }

        public char getGuess()
        {
            return guess;
        }

        public void setGuess(char guess)
        {
            this.guess = guess;
        }
    }

    class GuessBoard : Board
    {
        private int totalMisses;
        private int totalHits;

        public GuessBoard()
        {
            initialise(null);
            totalHits = 0;
            totalMisses = 0;
        }

        protected override char displayCell(Location location)
        {
            Object guess = getCell(location);
            return (guess == null ? '-' : ((Guess)guess).getGuess());
        }

        public void update(Location guessLocation, bool hasHit)
        {
            char guess;

            if (hasHit)
            {
                totalHits++;
                guess = 'x';
            }
            else
            {
                totalMisses++;
                guess = 'O';
            }

            setCell(guessLocation, new Guess(guess));
        }

        public int getTotalHits()
        {
            return totalHits;
        }

        public int getTotalMisses()
        {
            return totalMisses;
        }
    }

    class Game
    {
        private ShipBoard shipBoard;
        private GuessBoard guessBoard;

        public void play()
        {
            shipBoard = new ShipBoard();

            guessBoard = new GuessBoard();

            while(!hasWon() && !hasLost())
            {
                makeGuess();
            }
        }

        private Boolean hasLost()
        {
            if (guessBoard.getTotalMisses() > 20)
            {
                Console.WriteLine("You've lost :(");
                shipBoard.showBoard();
                Console.ReadLine();
                return true;
            }

            return false;
        }

        private Boolean hasWon()
        {
            if (guessBoard.getTotalHits() == shipBoard.getTotalShips())
            {
                Console.WriteLine("You've won :)");
                Console.ReadLine();
                return true;
            }

            return false;
        }

        private void makeGuess()
        {
            Console.Clear();
            guessBoard.showBoard();

            bool guessValid = true;
            Location guessLocation;

            do
            {
                Console.WriteLine(guessValid ? "Enter a location (e.g. A1):    ↩
                  " : "Please enter a valid location: ");
                String guess = Console.ReadLine().ToUpper();

                if (guess == "")
                {
                    System.Environment.Exit(1);
                }

                if (guess.Length != 2)
                {
                    guessValid = false;
                }
                else
                {
                    try
```

```csharp
                    {
                        guessLocation = new Location(guess);
                        guessBoard.update(guessLocation, shipBoard.hasHit(guessLocation));
                        guessValid = true;
                    }
                    catch
                    {
                        guessValid = false;
                    }
                }
            } while (guessValid == false);
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            new Game().play();
        }
    }
}
```