```
declare a global boolean winnerDeclared to determine if the game should end
declare a global Player human for the human player
declare a global Player computer for the AI
declare a global boolean playerTurn to determine if it's the player's turn
declare a volatile global boolean targeting to tell when the main thread should sleep in order to wait for the human player to fire their shot
declare a global HashSet<Pair> nextTargets for the expert AI to keep track of the enemy board
declare global booleans for each ship placed: destroyerPlaced, submarinePlaced, cruiserPlaced, battleshipPlaced, carrierPlaced
declare a volatile global boolean allPlaced which determines if the player is ready after placing their ships
declare two global Pair objects head, tail to keep track of the head and tail of ship placement
declare two volatile global booleans headPicked, tailPicked to determine if the user's selection is the head or tail of the placement

class members of "Player"{
	String playerName to keep track of who this player is
	Ship destroyer
	Ship submarine
	Ship cruiser
	Ship battleship
	Ship carrier
	int sunkenShips to keep track of how many ships have been sunk
	int shotsTaken to keep track of how many shots the player has taken
	int hits to keep track of how many hits the player has made
	int misses to keep track of how many misses the player has made

	char[10][10] playerGrid to keep track of the player's grid

	void method randomlyPlace(){
		do{
		choose a random point on the AI’s grid as the end of the carrier (two random numbers, 1-10)
		choose a random orientation (up down left or right) (random number from 1-4)
		check if the ship would cross with other ships or would be out of bounds
		if(validPlacement(computer, 'a')){
					carrierPlaced = true;
		}while(carrierPlaced==false);

		do{
		choose a random point on the AI’s grid as the end of the battleship (two random numbers, 1-10)
		choose a random orientation (up down left or right) (random number from 1-4)
		check if the ship would cross with other ships or would be out of bounds
		if(validPlacement(computer, 'b')){
					battleshipPlaced = true;
		}while(battleshipPlaced==false);
```

```
        do{
        choose a random point on the AI's grid as the end of the cruiser (two random numbers, 1-10)
        choose a random orientation (up down left or right) (random number from 1-4)
        check if the ship would cross with other ships or would be out of bounds
        if(validPlacement(computer, 'c')){
                                cruiserPlaced = true;
        }while(cruiserPlaced==false);

        do{
        choose a random point on the AI's grid as the end of the submarine (two random numbers, 1-10)
        choose a random orientation (up down left or right) (random number from 1-4)
        check if the ship would cross with other ships or would be out of bounds
        if(validPlacement(computer, 's')){
                                submarinePlaced = true;
        }while(submarinePlaced==false);

        do{
        choose a random point on the AI's grid as the end of the destroyer (two random numbers, 1-10)
        choose a random orientation (up down left or right) (random number from 1-4)
        check if the ship would cross with other ships or would be out of bounds
        if(validPlacement(computer, 'd')){
                                destroyerPlaced = true;
        }while(destroyerPlaced==false);

        carrierPlaced = false;
        battleshipPlaced = false;
        cruiserPlaced = false;
        submarinePlaced = false;
        destroyerPlaced = false;

        File printTo = new File("shipLocation.txt");
        PrintWriter output = new PrintWriter(printTo);
        for (int r=0;r<10;r++){
                for(int c=0;c<10;c++){
                        output.print(playerGrid[r][c]);
                }
                output.println();
        }
        output.close();
}

boolean method fire(Player opponent, Pair coordinates){
```

```
check if they've already fired here with: if (opponent.playerGrid[coordinates.row][coordinates.col] == 'X'||opponent.playerGrid[coordinates.row][coordinates.col] == 'O'){
        show a message to the user that they have already fired here
}
check if they missed with: else if(opponent.playerGrid[coordinates.row][coordinates.col] == '.'){
        show a message to the user that they missed
        set the coordinate on the opponent's grid to missed with: opponent.playerGrid[coordinates.row][coordinates.col] = 'O'
        to invert the playerTurn depending on if it's true or not, we can write: playerTurn = playerTurn? false: true
}
otherwise, they hit something: else{
        String shipHit to determine what ship the player hit
        boolean sinkingHit to determine if the hit sunk the ship
        switch (opponent.playerGrid[coordinates.row][coordinates.col]){
                case('d'):
                        shipHit = "destroyer"
                        opponent.destroyer.partsHit++
                        if(opponent.destroyer.partsHit==opponent.destroyer.shipSize){
                                sinkingHit = true
                                opponent.sunkenShips++
                        }
                        break
                case('s'):
                        shipHit = "submarine"
                        opponent.submarine.partsHit++
                        if(opponent.submarine.partsHit==opponent.submarine.shipSize){
                                sinkingHit = true
                                opponent.sunkenShips++
                        }
                        break
                case('c'):
                        shipHit = "cruiser"
                        opponent.cruiser.partsHit++
                        if(opponent.cruiser.partsHit==opponent.cruiser.shipSize){
                                sinkingHit = true
                                opponent.sunkenShips++
                        }
                        break
                case('b'):
                        shipHit = "battleship"
                        opponent.battleship.partsHit++
                        if(opponent.battleship.partsHit==opponent.battleship.shipSize){
                                sinkingHit = true
                                opponent.sunkenShips++
```

```
                                        }
                                        break
                                case('a'):
                                        shipHit = "carrier"
                                        opponent.carrier.partsHit++
                                        if(opponent.carrier.partsHit==opponent.carrier.shipSize){
                                                sinkingHit = true
                                                opponent.sunkenShips++
                                        }
                                        break
                        }
                        opponent.playerGrid[coordinates.row][coordinates.col] = 'X'
                        if(sinkingHit){
                                show a message that they've sunk the enemy's [shipHit]
                                if(opponent.sunkenShips == 5){
                                        winnerDeclared = true
                                        show a message that this player has won the match
                                }
                        }
                        else{
                                show a message that they've hit the enemy's [shipHit]
                        }
                        to invert the playerTurn depending on if it's true or not, we can write: playerTurn = playerTurn? false: true
                        return true so the AI knows it hit
                }
                return false so the AI knows it missed
        }
}

class members of "Pair"{
        int row, which tells the row of the coordinate
        int col, which tells the column of the coordinate
}

class members of "Ship"{
        int shipSize to how much space the ship should take up
        int partsHit to keep track of how many parts of the ship has been hit
}

boolean method coinToss(){
        use java Random class to generate a number of 0 or 1
        if(random number == 1){
```

```
                return true
        }
        else{
                return false
        }
}
boolean method validPlacement(Player player, char shipTypeChar){
        we can check if the placement is diagonal or out of bounds by: if((head.row-tail.row != 0 && head.col-tail.col != 0) || (coordinates.row < 0 || coordinates.row > 9 || coordinates.col < 0 || coordinates.col > 9)){
                return false
        }
        we check if the placement is horizontal by: if(head.row-tail.row == 0){
                int begin = min(head.col,tail.col) to determine where the for loop check should start
                int end = max(head.col,tail.col) to determine where the for loop should stop
                for(int i=begin;i<=max;i++){
                        we can check if the cell between the head and tail isn't an empty space by: if(player.playerGrid[head.row][i] != '.'){
                                return false
                        }
                }

                to update the 2d char array, we can: for(int i=begin;i<=max;i++){
                        player.playerGrid[head.row][i] = shipTypeChar
                }
                return true
        }
        otherwise, the placement is vertical: else {
                int begin = min(head.row,tail.row) to determine where the for loop check should start
                int end = max(head.row,tail.row) to determine where the for loop should stop
                for(int i=begin;i<=max;i++){
                        we can check if the cell between the head and tail isn't an empty space by: if(player.playerGrid[i][head.col] != '.'){
                                return false
                        }
                }
                to update the 2d char array, we can :for(int i=begin;i<=max;i++){
                        player.playerGrid[i][head.col] = shipTypeChar
                }
                return true
        }
}
actionperformed method for the 5 buttons for placing ships(Event e){
        String action = e.getActionCommand()

        If (action.equals(“Place Destroyer”)){
```

```
        if(!destroyerPlaced){
                headPicked = false
                tailPicked = false
                        while(!headPicked&&!tailPicked) {
                                try {
                                        Thread.sleep(200);
                                }
                                catch (InterruptedException e) {
                                }
                        }
                        if(validPlacement(human, 'd')){
                                destroyerPlaced = true;
                        }
                        else{
                                show a message that the placement is invalid
                        }
                }
                else{
                        show a message telling the player they have already placed the ship
                }
}
else if(action.equals(“Place Submarine”)){
        if(!submarinePlaced){
                headPicked = false
                tailPicked = false
                        while(!headPicked&&!tailPicked) {
                                try {
                                        Thread.sleep(200);
                                }
                                catch (InterruptedException e) {
                                }
                        }
                        if(validPlacement(human, 's')){
                                destroyerPlaced = true;
                        }
                        else{
                                show a message that the placement is invalid
                        }
                }
                else{
                        show a message telling the player they have already placed the ship
                }
```

```
}
else if(action.equals("Place Cruiser"){
	if(!cruiserPlaced){
		headPicked = false
		tailPicked = false
			while(!headPicked&&!tailPicked) {
				try {
					Thread.sleep(200);
				}
				catch (InterruptedException e) {
				}
			}
			if(validPlacement(human, 'c')){
				destroyerPlaced = true;
			}
			else{
				show a message that the placement is invalid
			}
		}
		else{
			show a message telling the player they have already placed the ship
		}
}
else if(action.equals("Place BattleShip"){
	if(!battleshipPlaced){
		headPicked = false
		tailPicked = false
			while(!headPicked&&!tailPicked) {
				try {
					Thread.sleep(200);
				}
				catch (InterruptedException e) {
				}
			}
			if(validPlacement(human, 'b')){
				destroyerPlaced = true;
			}
			else{
				show a message that the placement is invalid
			}
		}
		else{
```

```
                        show a message telling the player they have already placed the ship
                    }
            }
            else {
                    if(!carrierPlaced){
                        headPicked = false
                        tailPicked = false
                            while(!headPicked&&!tailPicked) {
                                try {
                                    Thread.sleep(200);
                                }
                                catch (InterruptedException e) {
                                }
                            }
                            if(validPlacement(human, 'a')){
                                destroyerPlaced = true;
                            }
                            else{
                                show a message that the placement is invalid
                            }
                        }
                        else{
                            show a message telling the player they have already placed the ship
                        }
            }

            if(destroyerPlaced && submarinePlaced && cruiserPlaced && battleshipPlaced && carrierPlaced){
                allPlaced = true;
            }
}


actionperformed method for buttons on the human player grid(Event e){
        get the coordinates from the JButton label and store as ButtonRow, ButtonCol
        we can check if the user is picking a new set of head and tail by checking: if(headPicked && tailPicked){
                headPicked = false;
                tailPicked = false;
        }

        if(!headPicked){
                head = new Pair{ButtonRow,ButtonCol};
                headPicked = true;
```

```
        }
        else{
                tail = new Pair{ButtonRow,ButtonCol};
                tailPicked = true;
        }
}

actionperformed method for buttons on the opponent's grid(Event e){
        we can check if they are allowed to fire currently with: if(targeting == true){
                get the coordinate of the button pressed on the computer player's grid from its JButton Label, the coordinates should be called ButtonRow and ButtonCol respectively
                human.fire(computer, Pair{ButtonRow,ButtonCol})
                targeting = false
        }
}

void method startGame(){
        initialize Player human
        initialize Player computer

        computer.randomlyPlace()
        user gets to pick between easy and expert difficulty with buttons
        user then gets 5 buttons labeled "Place [shipName]", which when pressed, lets them press 2 more buttons on the grid to determine the head and tail of the ship placement

        we can tell the main thread to sleep while we wait for the user to interact with the GUI with: while(!allPlaced) {
                try {
                        Thread.sleep(200);
                }
                catch (InterruptedException e) {
                }
        }

        if(coinToss returns true){
                playerTurn = true
        }

        while(!winnerDeclared){
                if(playerTurn is true){
                        targeting == true
                        we can tell the main thread to sleep while we wait for the user to fire their shot in the GUI with: while(targeting) {
                                try {
                                        Thread.sleep(200);
                                }
```

```
                                        catch (InterruptedException e) {
                                        }
                                }
                        }
                        else{
                                if(user picked easy){
                                        easyAITargeting()
                                }
                                else{
                                        expertAITargeting()
                                }
                        }
                }
}
void method easyAITargeting(){
        do{
                use java Random to generate two numbers between 0 to 9 inclusive called randomRow and randomCol
        }while(!validTargets(Pair{randomRow,randomCol}));
        computer.fire(human, Pair{randomRow, randomCol})
}

void method expertAITargeting(){
        The AI will randomly target the opponent until it strikes a hit, which causes it to find optimal targets
        declare two int: targetRow, targetCol to determine what coordinate it should fire on

        we can check if the AI has any optimal targets with: if(nextTargets.empty()){
                do{
                        use java Random to generate two numbers between 0 to 9 inclusive called targetRow and targetCol
                }while(!validTargets(Pair{targetRow, targetCol}));
        }
        otherwise it has optimal targets: else{
                initialize counter i to 0
                use an enhanced for loop to get the 'last' element it encounters: for(Pair p:nextTargets){
                        check if this is the last element with: if(i == nextTargets.size()){
                                set targetRow and targetCol to Pair p's row and col
                        }
                        i++
                }
                we can then remove the element from the list we just got by: nextTargets.remove(Pair{targetRow,targetCol})
        }
        the AI can check if it hit with: if(computer.fire(human, Pair{targetRow, randomCol} returns true){
                if(validTargets(Pair{targetRow+1, randomCol}){
```

```
                        nextTargets.add(Pair{targetRow+1, randomCol})
                }
                else if(validTargets(Pair{targetRow-1, randomCol}){
                        nextTargets.add(Pair{targetRow-1, randomCol})
                }
                else if(validTargets(Pair{targetRow, randomCol+1}){
                        nextTargets.add(Pair{targetRow, randomCol+1})
                }
                else if(validTargets(Pair{targetRow, randomCol-1}){
                        nextTargets.add(Pair{targetRow, randomCol-1})
                }
        }
}

boolean validTargets(Pair coordinates){
        if(coordinates.row < 0 || coordinates.row > 9 || coordinates.col < 0 || coordinates.col > 9|| human.playerGrid[randomRow][randomCol]!='X'|| human.playerGrid[randomRow][randomCol]!='O'){
                return false
        }
        return true
}
```