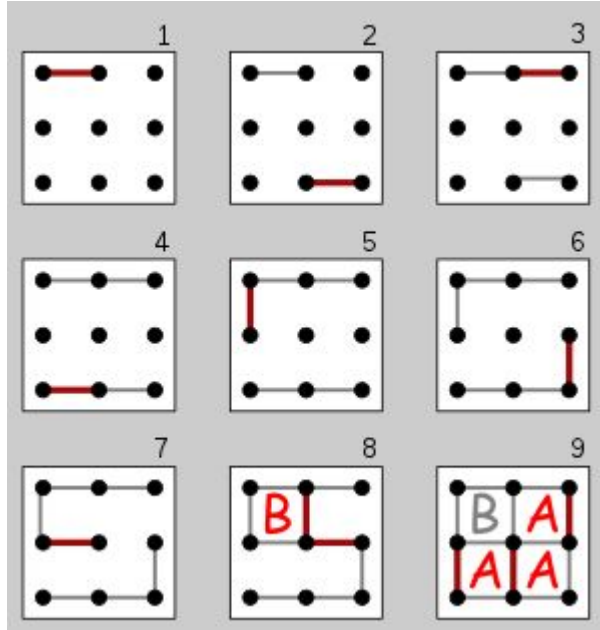




Dots and Boxes

Priyanshu Kumar





What is Dots
and Boxes?



01

UML

02

**Basic
Coding Principles**

Functions, Loops, If
statements, etc.

03

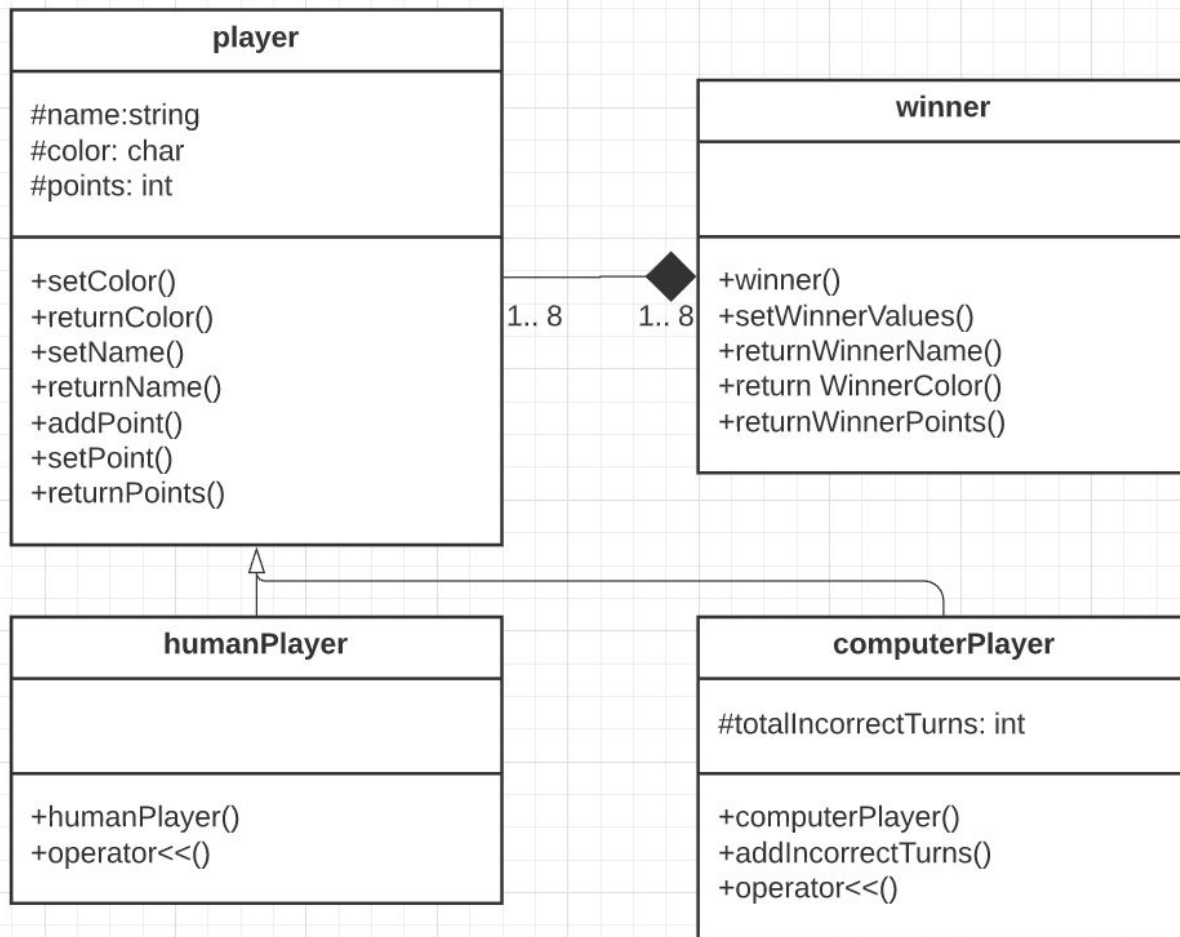
**Object Oriented
Programming**

Classes, Constructors,
Operator Overloading,
Inheritance, Object
Composition

04

Misc.

File IO, Multiple Files,
Exceptions



UML Diagram

Basic Coding Principles

Functions
If Statements
Loops
General Syntax

```

//Various Error Handling
while (errorInInput == true) {
    errorInInput = false;

    //Computer Input- Generates an input based via rand() and draws a line
    if (isComputer == true){ {... } }

    //Manual Input- Allows the user to enter an input and checks for valid user input
    if (isComputer == false){ {... } }

    //Changes From Char to Int
    for (int x = 0; x < 4; x++) {
        inputChar[x] = inputString[x];
        inputInt[x] = (int)inputChar[x] - 96;
    }
    cout << endl;

    //Prevents diagonal inputs
    if (errorInInput == false){ {... } }

    //Rearranges input to make sure the order is in a way that the program can process it
    if (inputInt[0] > inputInt[2]){ {... } }
    if (inputInt[1] > inputInt[3]){ {... } }

    //Check to see if the input is the same (Ex. dddd)
    if (inputInt[0] == inputInt[2] && inputInt[1] == inputInt[3]){ {... } } else {
        errorInInput = false;
    }

    //Check to see if input is too wide (aa,ac) or (ab,cb)
    if (errorInInput == false){ {... } }

    //Spot taken checker- Makes sure that the spot isn't taken already
    if (errorInInput == false){ {... } }

    //Sets the value of the globalInputInt
    if (errorInInput == false) {
        for (int x = 0; x < 4; x++) {
            globalInputInt[x] = inputInt[x];
        }
    }

    if (errorInInput == true && isComputer == true) {
        computerIncorrectInputs++; //Tracks to see how many times the computer generated an invalid input
        cout << "Computer entered invalid value. New value being entered: ";
    }

    else if (errorInInput == true && isComputer == false) {
        cout << "Invalid Input. Try again" << endl;
    }
}

```

Error Handling Section

- Uses for and while loops and if statements
- Ensures that a valid input is received
- Checks input against a 2d vector global integer array to see if spot was taken or not

A	B	C	
.	.	.	a
	r		
.	.	.	b
.	.	.	c

```

[0][1][0][0][0]
[1][0][1][0][1]
[0][1][0][0][0]
[0][0][0][0][0]
[0][0][0][0][0]

```

```
//Prints the int array for error checking if needed
//The programs relies on an int array to check to see which spots already have a line or which boxes can be made
void printGlobalIntArray() { ... }

//Creates the initial string for the board setup
//Also sets the value for globalArraySizeX/Y
void initBoardSetup() {
    int& arraySizeX = globalArraySizeX;
    int& arraySizeY = globalArraySizeY;
    string xLabel = "abcdefghijklmnopqrstuvwxyz";
    string yLabel = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    string& array = globalArray;

    cout << "How big of a board do you want? Minimum is 2 and maximum is 25" << endl;
    cin >> arraySizeX;
    while (!cin || arraySizeX < 2 || arraySizeX > 25) { ... }
    cin.clear();
    cin.ignore(INT_MAX, '\n');
    arraySizeY = arraySizeY;
    cout << "You have selected a " << arraySizeX << " by " << arraySizeY << " board" << endl;

    //Resizes Global Array
    //The Global Array must be set to 0 before assigning a new value. Otherwise the dimensions of the array are messed up
    globalIntArray.resize(0, vector<int>(0));
    globalIntArray.resize(arraySizeX * 2 - 1, vector<int>(arraySizeY * 2 - 1, 0));

    //Creates Array String
    for (int y = 0; y < arraySizeY; y++) { ... }
    array = array + " ";
    cout << endl;
    for (int y = 0; y < arraySizeY; y++) { ... }
}

//Displays the string array. Prints a long singular string in a way that the user can read
void displayArray() { ... }

//Asks the user if they would like to see instructions
void initStart() { ... }

//Checks for valid inputs and draws lines
void startGame(char playerColor, bool isComputer) { ... }

int main() {
```

Functions

- Most functions are void as a return statement is not needed
- Use of global variables to help return values between multiple functions

```
//Global Variables
int globalArraySizeX = 2;
int globalArraySizeY = 2;
int globalInputInt[4];
int globalComputerIncorrectInputs = 0;
string globalArray;
vector<vector<int>> globalIntArray(globalArraySizeY,
                                vector<int>(globalArraySizeX,0));
```

Object Oriented Programming

- Classes
- Constructors
- Operator Overloading
- Inheritance
- Object Composition
- Misc. Multiple Files


```
//Main Class: Players
class player {
protected:
    string name = ".";
    char color;
    int points = 0;
public:
    int returnPoints();
    char returnColor();
    void setColor(char color);
    void addPoint();
    void setPoints(int points);
    void setName(string name);
    friend ostream& operator<<(ostream& os, const player&);
    string returnName();
};
```

```
//Computer Player inherits from main class Players
class computerPlayer : public player {
protected:
    int totalIncorrectTurns = 0;
public:
    computerPlayer() {};
    friend ostream& operator<<(ostream& os, const computerPlayer&);
    void addIncorrectTurns(int incorrectTurns);
};
```

```
//Winner class uses composition. Every winner has a player (or players in ties)
class winner {
public:
    player winningPlayers;
    winner() {};
    void setWinnerValues(string nameWinner, int pointsWinner);
    string returnWinnerName();
    int returnWinnerPoints();
};
```

Classes

- Use of Inheritance and Composition to determine winners, computer players, and human players

- A default constructor was necessary for the use of "vector": Allow the use of dynamic arrays

- Classes were stored in "classesDB.h" header file and the rest of the program was stored in another file.

Classes

```
ostream& operator<<(ostream& os, const player& p) {  
    os << "Player " << p.name << " got " << p.points << " squares."  
    return os;  
}
```

-Overloaded the insertion operator to easily print out the values stored in the classes:
humanPlayers and computerPlayers

```
//Prints the scores for each individual player  
vector<winner> totalPlayers(hPlayersNum + cPlayersNum);  
cout << endl << "The overall scores are:-----" << endl;  
for (int x = 0; x < hPlayersNum; x++) {  
    cout << vecHumans[x] << endl;  
    totalPlayers[x].setWinnerValues(vecHumans[x].returnName(), vecHumans[x].returnPoints());  
}  
  
for (int x = 0; x < cPlayersNum; x++) {  
    cout << vecComputers[x] << endl;  
    totalPlayers[x+hPlayersNum].setWinnerValues(vecComputers[x].returnName(), vecComputers[x].returnPoints());  
}
```

Miscellaneous

File IO
Exceptions
Vector/Colors

File IO

- Used to print instructions to the user if needed
- Was extensively used when making designing the string that would print the dots and boxes board to the user

```
if (instructions == 1) { // Prints Directions to File
    cout << "Directions have been printed to the text file: directions.txt" << endl;
    ofstream fout2("directions.txt");
    fout2 << "Players take turns drawing horizontal or vertical lines between two adjacent points." << endl;
    fout2 << "If a player completes the fourth wall of the box, they can claim it as their own and then take another turn." << endl;
    fout2 << "The person with the most amount of boxes at the end wins the game." << endl;
    fout2.close();
}
else if (instructions == 0) {
    cout << "Displaying no instructions" << endl;
}
```

A B C D

. [32m__ [0m. [32m__ [0m. [32m__ [0m. a
 [31m| [0m a [31m| [0m a [31m| [0m x [31m| [0m
 . [31m__ [0m. [32m__ [0m. [32m__ [0m. b
 [31m| [0m a [31m| [0m a [32m| [0m x [32m| [0m
 . [31m__ [0m. [32m__ [0m. [31m__ [0m. c
 [32m| [0m x [32m| [0m a [31m| [0m x [32m| [0m
 . [31m__ [0m. [31m__ [0m. [32m__ [0m. d

A B C D

. [30m__ [0m. [30m__ [0m. [30m__ [0m. a
 [30m| [0m [30m| [0m [30m| [0m [30m| [0m
 . [30m__ [0m. [30m__ [0m. [30m__ [0m. b
 [30m| [0m [30m| [0m [30m| [0m [30m| [0m
 . [30m__ [0m. [30m__ [0m. [30m__ [0m. c
 [30m| [0m [30m| [0m [30m| [0m [30m| [0m
 . [30m__ [0m. [30m__ [0m. [30m__ [0m. d

A	B	C	D	
a	a	x		a
a	a	x		b
x	a	x		c
				d

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
c	a	a	d	d	d	a	a	a	c	c	b	b	b	b	b	d	c	b	c	b	b	b	b	b
a	a	a	d	d	c	a	a	a	b	c	c	c	c	c	b	b	c	c	c	b	b	c	c	c
a	d	d	b	d	d	d	a	a	a	a	d	d	c	c	d	b	b	d	d	a	b	c	c	c
a	d	d	d	d	d	d	a	d	d	a	d	d	d	c	d	a	b	b	d	d	b	b	a	a
a	c	c	d	d	d	b	a	a	d	d	c	a	d	d	d	b	c	b	b	c	c	b	a	a
c	c	c	a	d	d	b	c	d	d	a	c	c	b	c	c	b	c	b	b	c	c	c	a	a
b	c	a	a	d	d	a	a	a	d	a	d	d	d	c	c	a	a	c	a	d	c	c	d	d
a	c	a	a	a	a	a	a	a	d	d	d	a	d	c	c	b	b	c	c	c	c	d	d	d
a	a	a	a	a	a	d	c	c	d	d	d	b	b	b	b	b	d	c	c	c	a	a	a	a
b	b	a	b	c	c	a	a	a	d	d	c	c	a	a	a	a	d	c	c	c	b	d	d	d
b	b	d	d	c	c	d	d	a	d	d	d	d	a	c	c	a	d	b	a	a	a	d	d	d
a	d	d	d	a	a	a	c	a	c	b	d	d	d	c	c	a	a	b	d	b	d	c	c	c
b	b	b	a	a	a	d	d	a	a	a	a	d	d	d	d	d	d	b	b	a	a	a	a	a
b	c	d	d	d	a	c	c	a	a	c	a	a	b	b	d	d	a	b	b	b	a	a	b	a
c	d	d	d	d	a	a	d	a	b	a	a	a	b	b	a	a	b	b	d	d	a	b	a	a
d	d	c	c	d	a	a	b	b	b	a	a	a	a	a	a	a	a	b	d	a	a	a	b	c
c	c	c	c	c	a	d	d	b	b	b	b	a	b	b	b	b	b	b	d	c	a	a	c	c
c	b	b	c	c	a	d	d	a	a	b	b	b	b	c	c	c	a	a	c	d	d	c	c	c
d	a	c	d	a	a	c	c	a	b	b	c	b	c	c	c	c	b	b	c	d	d	b	c	c
a	a	d	d	d	a	a	a	c	c	b	b	a	a	c	c	b	b	d	c	d	d	a	d	d
d	d	d	d	a	a	a	d	d	d	b	b	a	a	c	c	b	b	d	a	a	d	d	d	d
d	d	d	d	c	a	a	b	b	b	c	c	b	a	a	b	a	a	a	a	a	a	d	d	d
d	d	d	d	c	c	c	b	a	d	a	a	a	a	a	a	a	a	d	d	b	b	a	d	c
c	d	d	d	d	a	c	b	a	d	a	a	d	b	b	a	a	d	b	b	d	d	d	d	d

The overall scores are:-----

Computer player a got 172 squares and entered 5086 invalid inputs

Computer player b got 116 squares and entered 5155 invalid inputs

Computer player c got 126 squares and entered 5372 invalid inputs

Computer player d got 162 squares and entered 5197 invalid inputs

```

__|0m.   e   |31m|0m a |31m|0m c |33m|0m c |34m|0m d |
__|0m.0|31m__|0m.0|34m__|0m.0|34m__|0m.0|33m__|0m.0|33m__|
a |34m|0m a |31m|0m a |31m|0m d |32m|0m d |34m|0m d |
__|0m.0|33m__|0m.0|31m__|0m.0|32m__|0m.0|32m__|0m.0|32m__|
c |32m|0m c |33m|0m a |34m|0m d |34m|0m b |31m|0m a |
__|0m.0|33m__|0m.0|31m__|0m.0|33m__|0m.0|34m__|0m.0|34m__|
b |31m|0m a |34m|0m      .|33m__|0m.0|32m__|0m.0|33m__|0
|31m|0m c |32m|0m c |31m|0m d |31m|0m a |31m|0m a |3
__|0m.0|34m__|0m.0|34m__|0m.0|33m__|0m.0|33m__|0m.0|34m__|0
|31m|0m b |34m|0m b |33m|0m c |32m|0m b |32m|0m c |3
__|0m.0|33m__|0m.0|33m__|0m.0|32m__|0m.0|32m__|0m.0|33m__|0
|31m|0m a |34m|0m a |32m|0m a |33m|0m a |34m|0m d |3
__|0m.0|34m__|0m.0|32m__|0m.   x   |34m|0m c |32m|0m d |34

```

Ln 1, Col 15362

-Print time: 26s

-Print time with arrays: 21min 43s

~18,450,000 characters

~14,000 characters/s

Exceptions

- Part of error handling involved checking to see if the input of array of characters had any integers in it
- stoi() is used to see if the input starts with an integer or contains only integers
- Other arguments result in an exception which is caught

```
try {  
    intsInString = stoi(inputString);  
    if (intsInString > 0) { //Checks to see if the string only consists of numbers  
        isInputInvalid = true;  
    }  
}  
catch (invalid_argument&) { //Catches an exception if the inputString contains no integers  
}
```

Things Included That Were Not Learned in Class

Vector

- Allows the use of dynamic arrays for things such as classes (The number of players and thus classes may vary)
- Can make a 2D vector array as well

Colors

- ANSI Escape Color Codes

```
cout << "\x1B[30mColor\x033[0m " << endl;  
cout << "\x1B[31mColor\x033[0m " << endl;  
cout << "\x1B[32mColor\x033[0m " << endl;  
cout << "\x1B[33mColor\x033[0m " << endl;  
cout << "\x1B[34mColor\x033[0m " << endl;  
cout << "\x1B[35mColor\x033[0m " << endl;  
cout << "\x1B[36mColor\x033[0m " << endl;  
cout << "\x1B[37mColor\x033[0m " << endl;
```

black

Color
Color
Color
Color
Color
Color
Color