Caleb Starkey, Dylan Stump

Player.h

```
struct Position {
	int row;
	int col;

	bool operator==(const Position &other) {
		return row == other.row && col == other.col;
	}
};

class Player {
public:
	Player(const std::string name, const bool is_human);

	string get_name() const {return name_; }

	int get_points() const {return points_; }

	Position get_position() const {return pos_; }

	bool is_human() const {return is_human_; }

	bool hasTreasure() const {return has_Treasure_; }

	bool isDead() const {return isDead_; }

	void ChangePoints(const int x);

	void SetPosition(Position pos);

	void setHasTreasure();

	void setIsDead(bool isdead);

	string ToRelativePosition(Position other);

	string Stringify();

private:
	string name_;
	int points_;
	Position pos_;
	bool is_human_;
	bool has_Treasure_;
	bool isDead_;

}; // class Player
```

Game.h

```
class Board {
public:
	Board();

	int get_rows() const {return 10; }
	int get_cols() const {return 10; }

	SquareType get_square_value(Position pos) const;

	void SetSquareValue(Position pos, SquareType value);

	vector<Position> GetMoves(Player *p);

	bool MovePlayer(Player *p, Position pos, vector<Player*> enemylist);

	bool MoveEnemy(Player *p, Position pos);

	friend ostream& operator<<(ostream& os, const Board &b);

private:
	SquareType arr_[10][10];
	int rows_; // might be convenient but not necessary
	int cols_;
};  // class Board

class Game {
public:
	Game();

	void NewGame(Player *human,vector<Player*> enemylist, const int
enemies);

	void TakeTurn(Player *p,vector<Player*> enemylist);

	void TakeTurnEnemy(Player *p);

	bool IsGameOver(Player *p);

	bool CheckifdotsOver();

	string GenerateReport(Player *p);

	friend ostream& operator<<(ostream& os, const Game &m);

private:
	Board *board_;
	vector<Player *> players_;
	int turn_count_;
	int dots_count_;
	bool GameOver;
};  // class Game
```

1) Annotating Player.h and Game.h:

a) Draw a square around the constructors for the Player, Board, and Game objects.

b) Draw a circle around the fields (class attributes) for the Player, Board, and Game objects.

c) Underline any methods that you think should not be public. (Briefly) Explain why you think that they should not be public.

2) Critiquing the design of the “pacman" game:

a) Methods: should do 1 thing and do it well. They should avoid long parameter lists and lots of boolean flags. Which, if any, methods does your group think are not designed well? Is there a method that you think is a good example of being well-designed? which?

Game constructor/ NewGame felt repetitive. Gameplay loop ended up mostly in NewGame but could have been in Game/ main.cpp if it was handled differently.

Move player and take turn were nicely designed.

b) Fields: should be part of the inherent internal state of the object. Their values should be meaningful throughout the object's life, and their state should persist longer than any one method. Which, if any, fields does your group think should not be fields? Why not? What is an example of a field that definitely should be a field? why?

Turn count was kind of pointless. GameOver boolean is redundant due to Is game over. players_ vector is redundant if your always passing enemy list

Previous squaretype for board square.

c) Fill in the following table. Briefly justify whether or not you think that a class fulfills the given trait.

Trait	Player	Board	Game
cohesive (one single abstraction)	Yes, all methods/traits apply there was nothing random	Mostly, move player/move enemy could have been in Game though	Yes, all the methods/traits applied
complete (provides a complete interface)	Mostly, we all added one or two different fields for convenience	Yes, also added a few fields that weren't strictly necessary but convenient	Yes, if anything there were to many bits
clear (the interface makes sense)	Yes	Yes	Some of the methods/fields felt extra and made Game feel a little muddled
convenient (makes things simpler in the long run)	Yes	Yes	Yes
consistent (names, parameters, ordering, behavior should be consistent)	Some of the method names used underscores others didn't	Yes	Yes