Tiana Koundakjian- 260949364

Assignment 2 Report

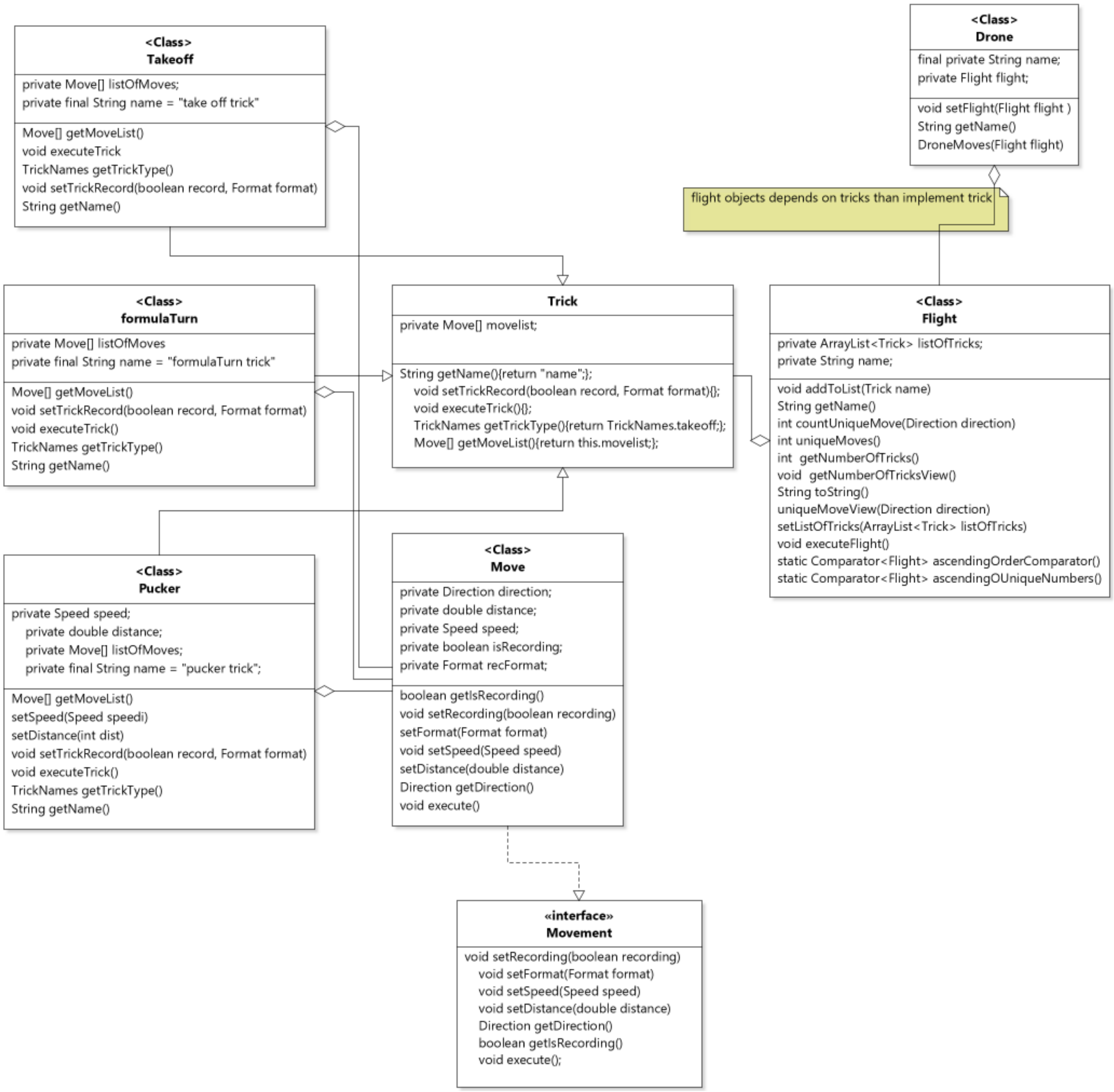
Encapsulation in this project is applied by private access modifiers along with getters and setters. In addition, assert statements were used to prevent Null Exception errors and negative distance value.

In terms of design principles seen in class, the movement interface is implemented by the concrete move class. Polymorphism is also applied through the Trick parent class. This class is implemented by 3 child classes. Two classes which implement the Comparator interface were used to allow the client to compare flights in ascending order of number of tricks, or number of unique moves.

Trade offs: Could have made Trick an interface but a parent child relationship between classes was optimal as the Flight takes an ArrayList<Trick>.

Different options: Instead of utilizing switch statements also known as an anti-pattern, using multiple enums was the favorable design choice to make.

Finally to satisfy the required problem statements, my design assumes that the Drone executes the Flight and can execute one flight at a time. The client can choose to record a trick in the flight which will record all the moves in the trick instead of each move individually. This is to prevent information leaking. Client does not have access to the list of moves in the trick.



<Class>
Takeoff

private Move[] listOfMoves;
private final String name = "take off trick"

Move[] getMoveList()
void executeTrick
TrickNames getTrickType()
void setTrickRecord(boolean record, Format format)
String getName()

<Class>
Drone

final private String name;
private Flight flight;

void setFlight(Flight flight)
String getName()
DroneMoves(Flight flight)

flight objects depends on tricks than implement trick

<Class>
formulaTurn

private Move[] listOfMoves
private final String name = "formulaTurn trick"

Move[] getMoveList()
void setTrickRecord(boolean record, Format format)
void executeTrick()
TrickNames getTrickType()
String getName()

Trick

private Move[] movelist;

String getName(){return "name";};
void setTrickRecord(boolean record, Format format){};
void executeTrick(){};
TrickNames getTrickType(){return TrickNames.takeoff;};
Move[] getMoveList(){return this.movelist;};

<Class>
Flight

private ArrayList<Trick> listOfTricks;
private String name;

void addToList(Trick name)
String getName()
int countUniqueMove(Direction direction)
int uniqueMoves()
int getNumberOfTricks()
void getNumberOfTricksView()
String toString()
uniqueMoveView(Direction direction)
setListOfTricks(ArrayList<Trick> listOfTricks)
void executeFlight()
static Comparator<Flight> ascendingOrderComparator()
static Comparator<Flight> ascendingOUniqueNumbers()

<Class>
Pucker

private Speed speed;
private double distance;
private Move[] listOfMoves;
private final String name = "pucker trick";

Move[] getMoveList()
setSpeed(Speed speedi)
setDistance(int dist)
void setTrickRecord(boolean record, Format format)
void executeTrick()
TrickNames getTrickType()
String getName()

<Class>
Move

private Direction direction;
private double distance;
private Speed speed;
private boolean isRecording;
private Format recFormat;

boolean getIsRecording()
void setRecording(boolean recording)
setFormat(Format format)
void setSpeed(Speed speed)
setDistance(double distance)
Direction getDirection()
void execute()

«interface»
Movement

void setRecording(boolean recording)
void setFormat(Format format)
void setSpeed(Speed speed)
void setDistance(double distance)
Direction getDirection()
boolean getIsRecording()
void execute();