Documentation

Project Description:

Simulate a simplified Capitaly game. There are some players with different strategies, and a cyclical board with several fields. Players can move around the board, by moving forward with the amount they rolled with a dice. A field can be a property, service, or lucky field. A property can be bought for 1000, and stepping on it the next time the player can build a house on it for 4000. If a player steps on a property field which is owned by somebody else, the player should pay to the owner 500, if there is no house on the field, or 2000, if there is a house on it. Stepping on a service field, the player should pay to the bank (the amount of money is a parameter of the field). Stepping on a lucky field, the player gets some money (the amount is defined as a parameter of the field). There are three different kind of strategies exist. Initially, every player has 10000. Greedy player: If he steps on an unowned property, or his own property without a house, he starts buying it, if he has enough money for it. Careful player: he buys in a round only for at most half the amount of his money. Tactical player: he skips each second chance when he could buy. If a player has to pay, but he runs out of money because of this, he loses. In this case, his properties are lost, and become free to buy.

Read the parameters of the game from a text file. This file defines the number of fields, and then defines them. We know about all fields: the type. If a field is a service or lucky field, the cost of it is also defined. After the these parameters, the file tells the number of the players, and then enumerates the players with their names and strategies. In order to prepare the program for testing, make it possible to the program to read the roll dices from the file.

Print out what we know about each player after a given number of rounds (balance, owned properties).

User Doc:

To use this program, you need to create two text files and place it in the project folder. Data in the text files should be like:

File 1:

First Line – Number of Fields (n)

The next n lines should contain the details of the Fields:

First Character – Type of Field

- S -> Service
- L -> Lucky
- P -> Property

Type of Field can only be one of these three.

The Second Character after space will be the cost of Field (Integer type.)

After n lines of fields,

Number of Players(m)

The next m lines should contain the details of the Players:

First Character – Strategy of Player

- G -> Greedy
- C -> Careful
- T -> Tactical

Type of Player can only be one of these three.

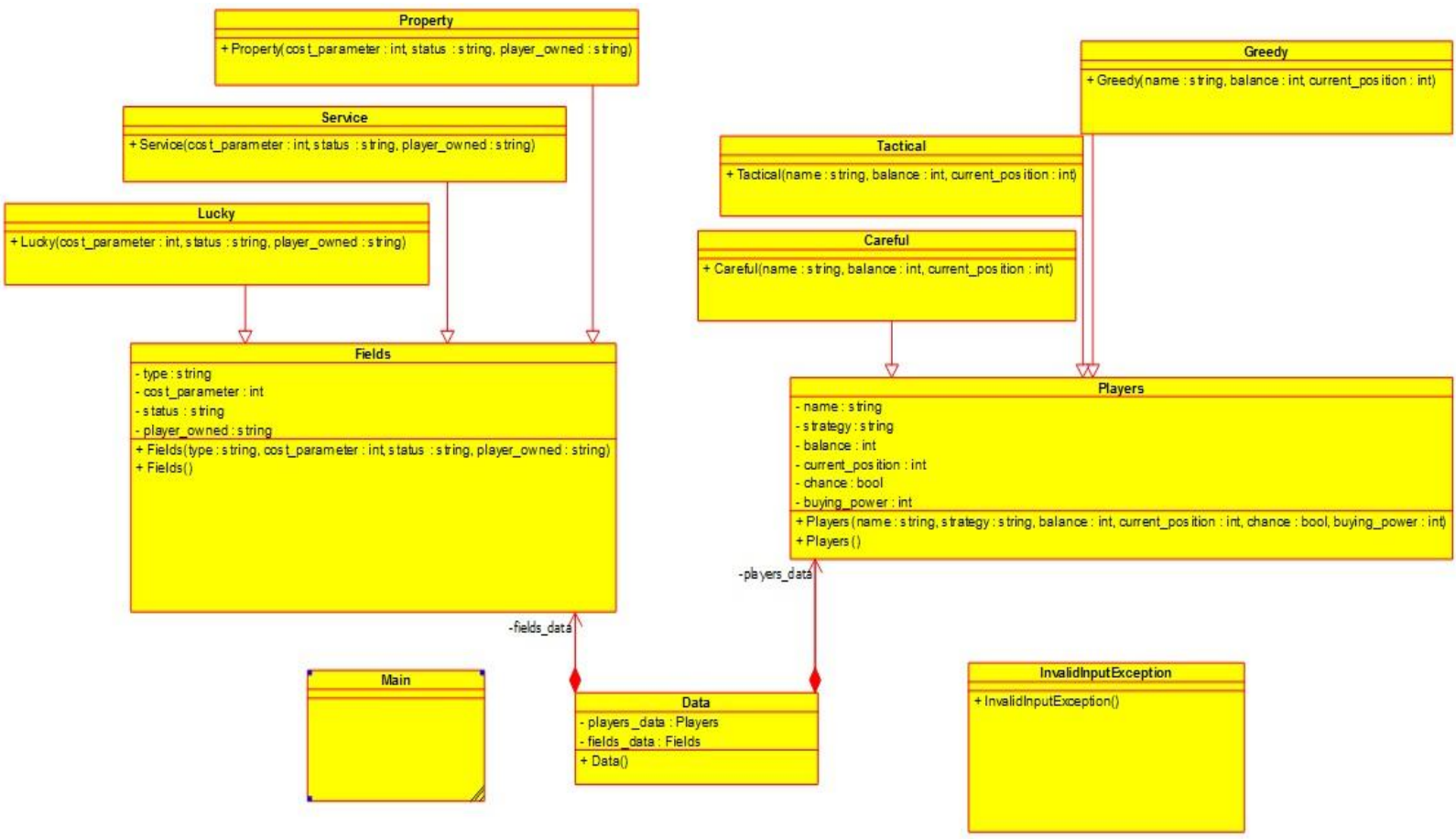
The second character after space will be the name of the player.

File 2:

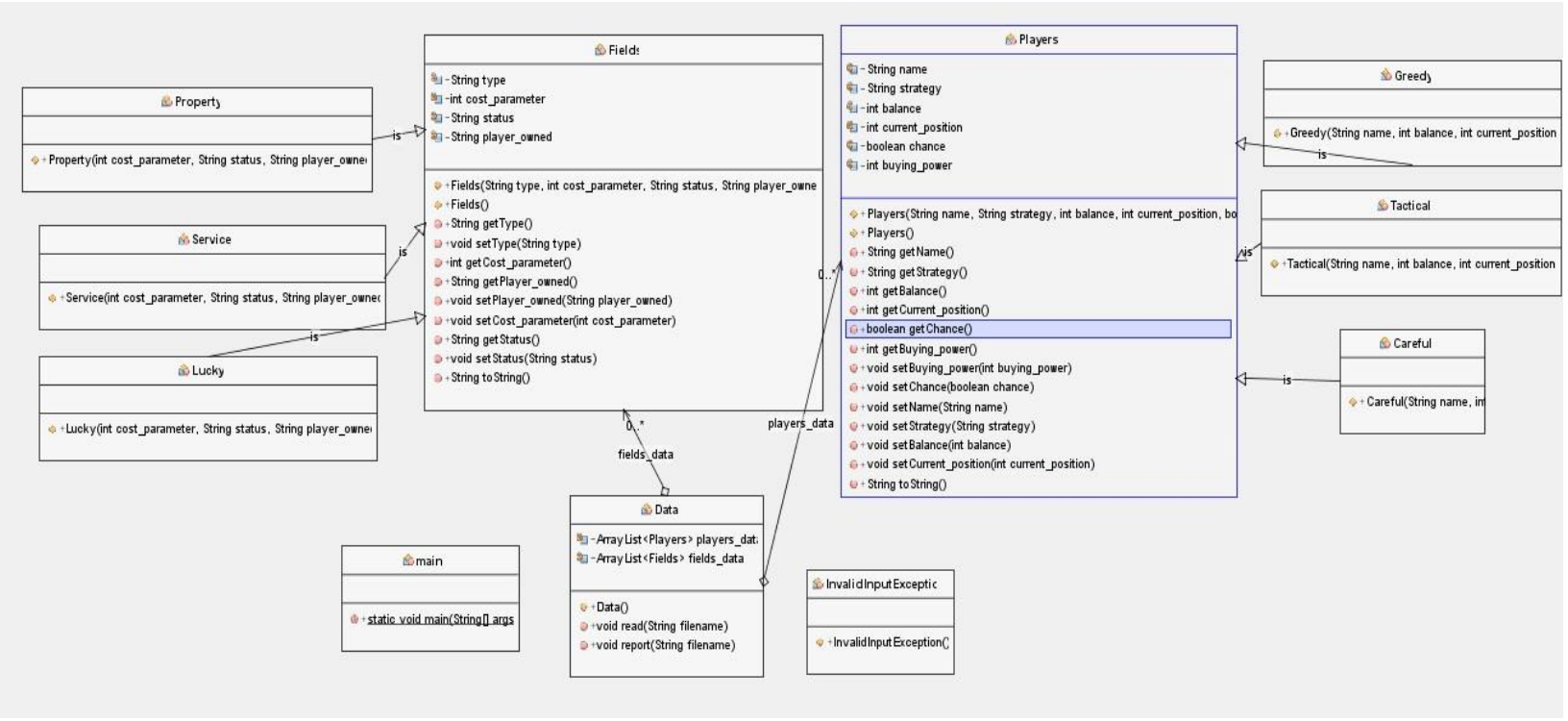
It contains the roll dices.

UML Diagram:

Using Umbrello



Using Netbeans



Implemented Methods:

Method of input class:

```
public void read(String filename) throws FileNotFoundException,
InvalidInputException

    {

    Scanner reader = new Scanner(new File(filename));

        if (reader.hasNextInt())

        {

            int num_of_fields;

            String type;
```

```
int cost;


num_of_fields = reader.nextInt();
System.out.println(num_of_fields);
while(num_of_fields>0)

{

Fields allfields;

type=reader.next();
System.out.println(type);
switch(type)

{
    case "P":
        cost=reader.nextInt();
        System.out.println(cost);
        allfields= new Property(cost,"empty","none");
        break;
    case "S":
        cost=reader.nextInt();
        System.out.println(cost);
        allfields= new Service(cost,"empty","none");
        break;
    case "L":
        cost=reader.nextInt();
        System.out.println(cost);
        allfields= new Lucky(cost,"empty","none");
        break;
    default:
        throw new InvalidInputException();

}
num_of_fields--;
fields_data.add(allfields);
}

if(reader.hasNextInt())
{
int num_of_players;
String name;
String strategy;
num_of_players = reader.nextInt();
System.out.println(num_of_players);
while(num_of_players>0)
{
    Players allplayers;

    strategy=reader.next();
    System.out.println(strategy);
    switch(strategy)
```

```
                {
                    case "G":
                        name=reader.next();
                        System.out.println(name);
                        allplayers=new Greedy(name,10000,0);
                        break;
                    case "C":
                        name=reader.next();
                        System.out.println(name);
                        allplayers=new Careful(name,10000,0);
                        break;
                    case "T":
                        name=reader.next();
                        System.out.println(name);
                        allplayers=new Tactical(name,10000,0);
                        break;
                    default:
                    throw new InvalidInputException();

                }

                num_of_players--;

                players_data.add(allplayers);
            }
            }
        }
        reader.close();
    }
```

This read method will read data about Fields and Players and put them into an arraylist.

```
public void report(String filename) throws FileNotFoundException,
InvalidInputException
    {
        Scanner dice_cnt = new Scanner(new File(filename));
        int total_players;
        int total_fields;
        int current_player=0;
        int dice;
        int previous_position;
        int current_position;
        boolean chance;
        //boolean operated;
        Players player=new Players();
        Fields field=new Fields();
        int bank=0;
        int max_players=players_data.size();
```

```
        while (dice_cnt.hasNextInt() && players_data.size()>1)
        {
         chance=true;
         dice=dice_cnt.nextInt();
         //System.out.println(dice);
         total_players=players_data.size();
         total_fields=fields_data.size();
         player=players_data.get(current_player%total_players);
         previous_position=player.getCurrent_position();
         //System.out.println(previous_position);
         current_position=(previous_position+dice)%(fields_data.size()+1);
         //System.out.println(current_position);
         //field=fields_data.get(current_position-1);
         player.setCurrent_position(current_position);
         //System.out.println(dice+previous_position);
         if(player.getStrategy()=="C")
         {
             if((previous_position+dice)>total_fields)
             {
                 player.setBuying_power(player.getBalance()/2);
             }
         }

         if(dice==0)
         {
             players_data.set(current_player%total_players,player);
         }
         else
         {
             field=fields_data.get(current_position-1);

             if(field.getType()=="L")
             {

player.setBalance(player.getBalance()+field.getCost_parameter());

player.setBuying_power(player.getBuying_power()+field.getCost_parameter())
;
                 players_data.set(current_player%total_players,player);

             }

             else if(field.getType()=="S")
             {
                 if(player.getBalance()>=field.getCost_parameter())
                 {
                 player.setBalance(player.getBalance()-
field.getCost_parameter());
                 player.setBuying_power(player.getBuying_power()-
field.getCost_parameter());
                 bank=bank+field.getCost_parameter();
                 players_data.set(current_player%total_players,player);
```

```
                }
                else
                {
                    Fields prop=new Fields();
                    for(int i=0;i<fields_data.size();i++)
                    {
                        prop=fields_data.get(i);
                        if(prop.getPlayer_owned()==player.getName())
                        {
                            prop.setPlayer_owned("none");
                            prop.setStatus("empty");
                            fields_data.set(i,prop);
                        }

                    }
                    players_data.remove(current_player%total_players);
                    System.out.println("Player Lost:");
                    System.out.println(player);
                }
            }

            else if(field.getType()=="P")
            {
                if(field.getStatus()=="house" &&
field.getPlayer_owned()!=player.getName())
                {
                    if(player.getBalance()>=2000)
                    {
                        player.setBalance(player.getBalance()-2000);
                        player.setBuying_power(player.getBuying_power()-
2000);
                        String name_owner=field.getPlayer_owned();
                        int id_owner=0;
                        Players owner_data=new Players();

                        for(int i=0;i<players_data.size();i++)
                        {

                            owner_data=players_data.get(i);
                            if(owner_data.getName()==name_owner)
                            {
                                id_owner=i;
                                i=players_data.size();
                            }
                        }

owner_data.setBalance(owner_data.getBalance()+2000);

owner_data.setBuying_power(owner_data.getBuying_power()+2000);
                        players_data.set(id_owner, owner_data);

players_data.set(current_player%total_players,player);

                    }
```

```
                    else
                    {
                        String name_owner=field.getPlayer_owned();
                        int id_owner=0;
                        Players owner_data=new Players();

                        for(int i=0;i<players_data.size();i++)
                        {

                            owner_data=players_data.get(i);
                            if(owner_data.getName()==name_owner)
                            {
                                id_owner=i;
                                i=players_data.size();
                            }
                        }
                        Fields prop=new Fields();
                    for(int i=0;i<fields_data.size();i++)
                    {
                        prop=fields_data.get(i);
                        if(prop.getPlayer_owned()==player.getName())
                        {
                            prop.setPlayer_owned("none");
                            prop.setStatus("empty");
                            fields_data.set(i,prop);
                        }

                    }


owner_data.setBalance(owner_data.getBalance()+player.getBalance());

owner_data.setBuying_power(owner_data.getBuying_power()+player.getBalance(
));
                        players_data.set(id_owner, owner_data);

players_data.remove(current_player%total_players);
                        System.out.println("Player Lost:");
                        System.out.println(player);
                    }
                }

                else if(field.getStatus()=="owned" &&
field.getPlayer_owned()!=player.getName())
                {
                    if(player.getBalance()>=500)
                    {
                        player.setBalance(player.getBalance()-500);
                        player.setBuying_power(player.getBuying_power()-
500);
                        String name_owner=field.getPlayer_owned();
                        int id_owner=0;
                        Players owner_data=new Players();
```

```
                            for(int i=0;i<players_data.size();i++)
                            {

                                owner_data=players_data.get(i);
                                if(owner_data.getName()==name_owner)
                                {
                                    id_owner=i;
                                    i=players_data.size();
                                }
                            }

owner_data.setBalance(owner_data.getBalance()+500);

owner_data.setBuying_power(owner_data.getBuying_power()+500);
                            players_data.set(id_owner, owner_data);

players_data.set(current_player%total_players,player);

                        }
                        else
                        {
                            String name_owner=field.getPlayer_owned();
                            int id_owner=0;
                            Players owner_data=new Players();

                            for(int i=0;i<players_data.size();i++)
                            {

                                owner_data=players_data.get(i);
                                if(owner_data.getName()==name_owner)
                                {
                                    id_owner=i;
                                    i=players_data.size();
                                }
                            }
                            Fields prop=new Fields();
                        for(int i=0;i<fields_data.size();i++)
                        {
                            prop=fields_data.get(i);
                            if(prop.getPlayer_owned()==player.getName())
                            {
                                prop.setPlayer_owned("none");
                                prop.setStatus("empty");
                                fields_data.set(i,prop);
                            }

                        }


owner_data.setBalance(owner_data.getBalance()+player.getBalance());

owner_data.setBuying_power(owner_data.getBuying_power()+player.getBalance(
));
                            players_data.set(id_owner, owner_data);
```

```
players_data.remove(current_player%total_players);
                        System.out.println("Player Lost:");
                        System.out.println(player);
                    }
                }
                    else if(field.getStatus()=="owned" &&
field.getPlayer_owned()==player.getName())
                    {
                            if(player.getBalance()>=4000)
                            {
                            player.setBalance(player.getBalance()-4000);

player.setBuying_power(player.getBuying_power()-4000);
                            field.setStatus("house");
                            field.setPlayer_owned(player.getName());

players_data.set(current_player%total_players,player);
                            fields_data.set(current_position-1,field);
                            }
                    }

                    else if(field.getStatus()=="empty")
                    {
                        if(player.getBuying_power()>=1000)
                        {
                            if(player.getStrategy()=="T" &&
player.getChance()==true)
                            {
                                player.setBalance(player.getBalance()-
1000);

player.setBuying_power(player.getBuying_power()-1000);
                                player.setChance(false);
                                field.setStatus("owned");
                                field.setPlayer_owned(player.getName());

players_data.set(current_player%total_players, player);
                                fields_data.set(current_position-
1,field);
                            }
                            else if(player.getStrategy()=="T" &&
player.getChance()==false)
                            {
                                player.setChance(true);

players_data.set(current_player%total_players, player);
                            }

                            else if (player.getStrategy()!="T")
                            {
                                player.setBalance(player.getBalance()-
1000);
```

```
player.setBuying_power(player.getBuying_power()-1000);
                                    field.setStatus("owned");
                                    field.setPlayer_owned(player.getName());

players_data.set(current_player%total_players,player);
                                    fields_data.set(current_position-
1,field);
                                }
                            }
                        }
                    }
                }
                if(players_data.size()==total_players)
                {
                current_player=current_player+1;
                }
            }

        if(players_data.size()==1)
        {
            System.out.println("Winner of game:");
            Players win=new Players();
            win=players_data.get(0);
            System.out.println(win);
        }
        else
        {

         int i=1;
        for(Players a:players_data)
        {

            int j=0;
            System.out.println("Details of player"+"#"+i+":");
            System.out.println(a);
            System.out.println("Details of player"+"#"+i+"
properties"+":");
             for(Fields b:fields_data)
            {
                if(b.getPlayer_owned()==a.getName())
                {
                    System.out.println(b);
                    j=j+1;
                }
            }
             if(j==0)
             {
                 System.out.println("No properties owned!");
             }
             j=0;
            i=i+1;}}}
```

This report method will print the data what we know about each player.

Test Cases:

Sample Input 1:

File 1:

```
6
S 700
L 1000
P 900
P 900
L 1000
S 700
4
G Max
C Karan
T Frans
G Messi
```

File 2:

```
5
2
6
6
1
2
3
4
```

Sample Output 1:

```
Details of player#1:
Players{name=Max, strategy=G, balance=10300, current_position=6,
chance=false, buying_power=10300}
Details of player#1 properties:
No properties owned!
Details of player#2:
Players{name=Karan, strategy=C, balance=10000, current_position=4,
chance=false, buying_power=5000}
Details of player#2 properties:
Fields{type=P, cost_parameter=900, status=owned, player_owned=Karan}
Details of player#3:
Players{name=Frans, strategy=T, balance=10300, current_position=2,
chance=true, buying_power=10300}
Details of player#3 properties:
No properties owned!
Details of player#4:
Players{name=Messi, strategy=G, balance=8300, current_position=3,
chance=false, buying_power=8300}
Details of player#4 properties:
Fields{type=P, cost_parameter=900, status=owned, player_owned=Messi}
```

Sample Input 2:

File 1:

```
7
S 500
L 1200
L 1200
P 1000
S 500
P 1000
L 1200
6
C John
G Gabriel
C Ali
T Niki
G Ahmed
G Umaim
```

File 2:

```
1
2
3
4
5
6
6
5
4
3
2
1
```

Sample Output 2:

```
Details of player#1:
Players{name=John, strategy=C, balance=10700, current_position=7,
chance=false, buying_power=5700}
Details of player#1 properties:
No properties owned!
Details of player#2:
Players{name=Gabriel, strategy=G, balance=12400, current_position=7,
chance=false, buying_power=12400}
Details of player#2 properties:
No properties owned!
Details of player#3:
Players{name=Ali, strategy=C, balance=12400, current_position=7,
chance=false, buying_power=7400}
Details of player#3 properties:
No properties owned!
Details of player#4:
```

```
Players{name=Niki, strategy=T, balance=10200, current_position=7,
chance=false, buying_power=10200}
Details of player#4 properties:
Fields{type=P, cost_parameter=1000, status=owned, player_owned=Niki}
Details of player#5:
Players{name=Ahmed, strategy=G, balance=10700, current_position=7,
chance=false, buying_power=10700}
Details of player#5 properties:
No properties owned!
Details of player#6:
Players{name=Umaim, strategy=G, balance=10200, current_position=7,
chance=false, buying_power=10200}
Details of player#6 properties:
Fields{type=P, cost_parameter=1000, status=owned, player_owned=Umaim}
```

Sample Input 3:

File 1:

```
7
S 900
L 1500
P 1300
L 1500
S 900
P 1300
S 900
3
T David
G Mea
C Edi
```

File 2:

```
2
6
5
3
1
4
6
3
4
1
1
2
5
4
3
```

Sample Output 3:

```
Details of player#1:
Players{name=David, strategy=T, balance=10700, current_position=1,
chance=false, buying_power=10700}
Details of player#1 properties:
Fields{type=P, cost_parameter=1300, status=owned, player_owned=David}
Details of player#2:
Players{name=Mea, strategy=G, balance=8200, current_position=7, chance=false,
buying_power=8200}
Details of player#2 properties:
Fields{type=P, cost_parameter=1300, status=owned, player_owned=Mea}
Details of player#3:
Players{name=Edi, strategy=C, balance=7900, current_position=2, chance=false,
buying_power=4700}
Details of player#3 properties:
No properties owned!
```

Sample Input 4:

File 1:

```
6
P 700
S 600
L 1000
L 1000
S 600
P 700
4
C one
T two
G three
C four
T five
```

File 2:

```
2
3
6
5
4
1
3
5
2
6
1
1
1
2
2
```

Sample Output 4:

```
Details of player#1:
Players{name=one, strategy=C, balance=7300, current_position=2,
chance=false, buying_power=2850}
Details of player#1 properties:
Fields{type=P, cost_parameter=700, status=owned, player_owned=one}
Details of player#2:
Players{name=two, strategy=T, balance=12400, current_position=5,
chance=true, buying_power=12400}
Details of player#2 properties:
No properties owned!
Details of player#3:
Players{name=three, strategy=G, balance=9300, current_position=5,
chance=false, buying_power=9300}
Details of player#3 properties:
Fields{type=P, cost_parameter=700, status=owned, player_owned=three}
Details of player#4:
Players{name=four, strategy=C, balance=11400, current_position=4,
chance=false, buying_power=6700}
Details of player#4 properties:
No properties owned!
```

Sample Input 5:

File 1:

```
8
P 0
P 0
L 700
P 0
S 1100
S 1100
L 600
P 0
3
T Lilly
G Laila
C Laima
```

File 2:

```
6
5
4
3
2
1
```

Sample Output 5:

```
Details of player#1:
Players{name=Lilly, strategy=T, balance=8900, current_position=0,
chance=true, buying_power=8900}
Details of player#1 properties:
No properties owned!
Details of player#2:
Players{name=Laila, strategy=G, balance=9500, current_position=7,
chance=false, buying_power=9500}
Details of player#2 properties:
No properties owned!
Details of player#3:
Players{name=Laima, strategy=C, balance=7900, current_position=5,
chance=false, buying_power=2900}
Details of player#3 properties:
Fields{type=P, cost_parameter=0, status=owned, player_owned=Laima}
```

Sample Input 6:

File 1:

```
9
S 500
S 500
S 500
L 1500
L 1500
L 1500
P 100
P 100
P 100
8
C Adriel
G Bernice
G Bethany
T Charity
G Chloe
C Dinah
C Diana
T Elisha
```

File 2:

```
2
4
6
1
3
5
6
4
2
5
```

```
3
1
2
4
6
5
```

Sample Output 6:

```
Details of player#1:
Players{name=Adriel, strategy=C, balance=11000, current_position=4,
chance=false, buying_power=6000}
Details of player#1 properties:
No properties owned!
Details of player#2:
Players{name=Bernice, strategy=G, balance=12000, current_position=9,
chance=false, buying_power=12000}
Details of player#2 properties:
Fields{type=P, cost_parameter=100, status=owned, player_owned=Bernice}
Details of player#3:
Players{name=Bethany, strategy=G, balance=11000, current_position=9,
chance=false, buying_power=11000}
Details of player#3 properties:
No properties owned!
Details of player#4:
Players{name=Charity, strategy=T, balance=9000, current_position=2,
chance=true, buying_power=9000}
Details of player#4 properties:
No properties owned!
Details of player#5:
Players{name=Chloe, strategy=G, balance=11000, current_position=5,
chance=false, buying_power=11000}
Details of player#5 properties:
No properties owned!
Details of player#6:
Players{name=Dinah, strategy=C, balance=11000, current_position=9,
chance=false, buying_power=6000}
Details of player#6 properties:
No properties owned!
Details of player#7:
Players{name=Diana, strategy=C, balance=11000, current_position=2,
chance=false, buying_power=5250}
Details of player#7 properties:
No properties owned!
Details of player#8:
Players{name=Elisha, strategy=T, balance=11000, current_position=9,
chance=true, buying_power=11000}
Details of player#8 properties:
No properties owned!
```