CS 342 Report for Project 2

Code

Player.java

```
public class Player
{
    // DATA DICTIONARY
      private static final int INITIAL_MONEY = 1500;
      private static int dice = 0;           // dice value
    private final String token;              // token
    private int money;                    // how much money you have
    private BoardLocation location;          // the current player location
    private List<Property> properties;       // the properties we own
    private boolean bankrupt;             // is the person bankrupt

    public Player(String token, BoardLocation go)
    {
        this.money = INITIAL_MONEY;
        this.location = go;
        this.token = token;
        this.properties = new ArrayList<>();
    }

    public boolean buyLocation(Property property)
    {
        if ( property.getCost() <= money &&
             property.getOwner() == null )
        {
            properties.add(property);
            addMoney(-property.getCost());
            property.setOwner(this);
            return true;
        }
        return false;
    }

    public void sellLocation()
    {
      // TODO: not required!
    }

    public void move(int n)
    {
        for (int i = 0; i < n; i++)
        {
            location = location.getNext();
            if (location.getName().equals("Go"))
            {
                addMoney(200);
            }
        }

        if ( location instanceof Property &&
             ((Property)(location)).isOwned())
        {
            ((Property)(location)).collectRent(this);
        }
        else if(location instanceof CardSquare)
           ((CardSquare)(location)).reward(this);
    }

    public List<Property> getProperties()
    {
        return properties;
    }

    public void addMoney(int money)
    {
        setMoney(getMoney()+money);
    }

    public void transferMoneyTo(Player player, int money)
```

```
    {
        this.addMoney(-money);
        player.addMoney(money);
    }

    public int getMoney()
    {
        return money;
    }

    public void setMoney(int money)
    {
        if (this.money+money<0)
        {
            declareBankruptcy();
        }

        this.money = money;
    }

    private void declareBankruptcy()
    {
        this.bankrupt = true;
    }

    public BoardLocation getLocation()
    {
        return location;
    }

    public void setLocation(BoardLocation location)
    {
        this.location = location;
    }

    public boolean isBankrupt()
    {
        return bankrupt;
    }

      public int getDice()
      {
       return dice;
      }

      public String getToken()
      {
       return token;
      }

      @Override
      public String toString()
      {
       if(!bankrupt)
           return "Player: " + token + " has $" + money;
       else
           return "Player: " + token + " is broke :(";
      }
}
```

BoardLocation.java

```
public abstract class BoardLocation
{
    protected final String name;                            // piece name
    protected final int address;                            // the distance from go
    protected BoardLocation next;                    // the next place to move

    public BoardLocation(String name, int address)
    {
        this.name = name;
        this.address = address;
        this.next = null;
    }

    public abstract String[] getPossibleActions(Player player);

    public BoardLocation getNext()
    {
        return next;
    }

    public String getName()
    {
        return name;
    }

    @Override
    public String toString()
    {
        return "BoardLocation: " + name;
    }

    public static void Link(BoardLocation[] board)
    {
        for(int i=0;i<board.length - 1;i++)
            board[i].next = board[i+1];
        board[board.length-1].next = board[0];
    }
}
```

Property.java

```
public abstract class Property extends BoardLocation
{
    protected final int cost;                    // the cost of property
    protected Player owner;                  // who owns the  property


    public Property(String name, int address, int cost)
    {
        super(name,address);
        this.cost = cost;
        this.owner = null;
    }

    public abstract void collectRent(Player player);

    public int getCost()
    {
        return cost;
    }

    public Player getOwner()
    {
        return owner;
    }

    public void setOwner(Player owner)
    {
        this.owner = owner;
    }

    public boolean isOwned()
    {
        return owner != null;
    }

    @Override
    public String toString()
    {
       String player;

       player = owner == null ? "no one" : owner.getToken();
       return super.toString() + " costs $" + cost + " owned by " + player;
    }
}
```

Lot.java

```
public class Lot extends Property
{
    private final String color;                  // color of the board piece
    private final int[] rent;
    private int rentIndex;
    private final int improveCost;

    public Lot(String name, int address, int cost,
               String color, int improve, int[] rent)
    {
        super(name, address, cost);
        this.color = color;
        this.rent = rent;
        this.rentIndex = 0;
        this.improveCost = improve;
    }

    @Override
    public void collectRent(Player player)
    {
        int payment;

        payment = rent[rentIndex];

        player.transferMoneyTo(owner, payment);
    }

    public void improve()
    {
        if(owner != null && owner.getMoney() >= improveCost && rentIndex < rent.length - 1)
        {
            rentIndex++;
            owner.addMoney(-improveCost);
        }
    }

    public void diminish()
    {
        if(owner != null && rentIndex > 0)
        {
            rentIndex--;
            owner.addMoney(improveCost/2);
        }
    }

    @Override
    public String[] getPossibleActions(Player player)
    {
        return null;
    }

    @Override
    public String toString()
    {
        return super.toString() + " color: " + color;
    }
}
```

Railroad.java

```
public class RailRoad extends Property
{
      private RailRoad[] others;

      public RailRoad(String name, int address, int cost)
      {
        super(name, address, cost);
        others = new RailRoad[3];
      }

      public void setOthers(RailRoad other1, RailRoad other2, RailRoad other3)
      {
        others[0] = other1;
        others[1] = other2;
        others[2] = other3;
      }

      @Override
      public void collectRent(Player player)
      {
        int payment;

        payment = 25;
        for(RailRoad r : others)
            if(r.getOwner() == this.owner)
                payment *= 2;

        player.transferMoneyTo(owner, payment);
      }

      @Override
      public String[] getPossibleActions(Player player)
      {
        // TODO Auto-generated method stub
        return null;
      }


}
```

Utility.java

```
public class Utility extends Property
{
      private Utility other;

      public Utility(String name, int address, int cost)
      {
        super(name, address, cost);
      }

      @Override
      public void collectRent(Player player)
      {
        int rent;

        rent = other.owner == this.owner ? player.getDice() * 10 : player.getDice() * 4;
        player.transferMoneyTo(owner, rent);
      }

      public void setOther(Utility other)
      {
        this.other = other;
      }

      @Override
      public String[] getPossibleActions(Player player) {
        // TODO Auto-generated method stub
        return null;
      }

      @Override
      public String toString()
      {
        return super.toString();
      }

}
```

CornerSquare.java

```
public class CornerSquare extends BoardLocation
{
      public CornerSquare(String name, int address)
      {
        super(name, address);
      }

      @Override
      public String[] getPossibleActions(Player player)
      {
        // TODO Auto-generated method stub
        return null;
      }
}
```

TaxSquare.java

```
public class TaxSquare extends BoardLocation
{
	private final int tax;

	public TaxSquare(String name, int address, int tax)
	{
	  super(name, address);
	  this.tax = tax;
	}

	@Override
	public String[] getPossibleActions(Player player)
	{
	  // TODO Auto-generated method stub
	  return null;
	}

}
```

CardSquare.java

```
public class CardSquare extends BoardLocation
{
      public CardSquare(String name, int address)
      {
        super(name, address);
      }

      public int reward(Player player)
      {
        int amount;

        amount = (int)(Math.random()*401 - 200);
        player.addMoney(amount);

        return amount;
      }

      @Override
      public String[] getPossibleActions(Player player)
      {
        // TODO Auto-generated method stub
        return null;
      }

}
```

Test.java

```
public class Test
{
     public static void main(String[] args)
     {
       BoardLocation[] board;
       Player player1;
       Player player2;

       //init
       board = new BoardLocation[]
              {
                  new CornerSquare("Go", 0),
                  new Lot("MEDITERRANEAN AVE", 1, 60, "Dark Purple", 50,
                         new int[] { 2, 10, 30, 90, 160, 230}),
                  new CardSquare("Community Chest", 2),
                  new Lot("BALTIC AVE.", 3, 60, "Dark Purple", 50,
                         new int[] { 4, 20, 60, 180, 320, 450}),
                  new TaxSquare("Income Tax", 4, 200),
                  new RailRoad("READING RAILROAD", 5, 200),
                  new Lot("ORIENTAL AVE.", 6, 100, "Light Blue", 50,
                         new int[] { 6, 30, 90, 270, 400, 550}),
                  new CardSquare("Chance", 7),
                  new Lot("VERMONT AVE.", 8, 100, "Light Blue", 50,
                         new int[] { 6, 30, 90, 270, 400, 550}),
                  new Lot("CONNECTICUT AVE.", 9, 120, "Light Blue", 50,
                         new int[] { 8, 40, 100, 300, 450, 600}),
                  new CornerSquare("Jail/Just Visiting", 10),
                  new Lot("ST. CHARLES PLACE", 11, 140, "Light Purple", 100,
                         new int[] { 10, 50, 150, 450, 625, 750}),
                  new Utility("ELECTRIC COMPANY", 12, 150),
                  new Lot("STATES AVE.", 13, 140, "Light Purple", 100,
                         new int[] { 10, 50, 150, 450, 625, 750}),
                  new Lot("VIRGINIA AVE.", 14, 160, "Light Purple", 100,
                         new int[] { 12, 60, 180, 500, 700, 900}),
              };
       BoardLocation.Link(board);

       player1 = new Player("Car", board[0]);
       player2 = new Player("Boot", board[0]);

       for(BoardLocation b : board)
           System.out.println(b);

       System.out.println();

       player1.move(3);
       if(player1.buyLocation((Property) player1.getLocation()))
           System.out.println(player1 + " bought " + player1.getLocation());

       System.out.println(player2 + " moves 3");
       player2.move(3);
       System.out.println(player1 + " collected rent from " + player2);

       System.out.println(player1 + " moves 4");
       player1.move(4);
       System.out.println(player1 + " got rewarded");

       System.out.println(player1 + " buys 2 houses");
       ((Lot) player1.getProperties().get(0)).improve();
       ((Lot) player1.getProperties().get(0)).improve();
       System.out.println(player2 + " makes a cycle");
       player2.move(15);
       System.out.println(player1 + " collected rent from " + player2);

       ((Lot) player1.getProperties().get(0)).diminish();
       ((Lot) player1.getProperties().get(0)).diminish();
       System.out.println(player1 + " sold 2 houses");
     }
}
```