CS4710: Artificial Intelligence
Homework 3: Ticket to Ride
Fall 2015

Introduction:

For this assignment, you will be implementing an AI that plays the popular board game *Ticket to Ride*. The rules of the game are posted on Collab in case you are not yet familiar with the game. There will be a few simplification to the rules (see below). Additionally, a simulator framework will be given to you.

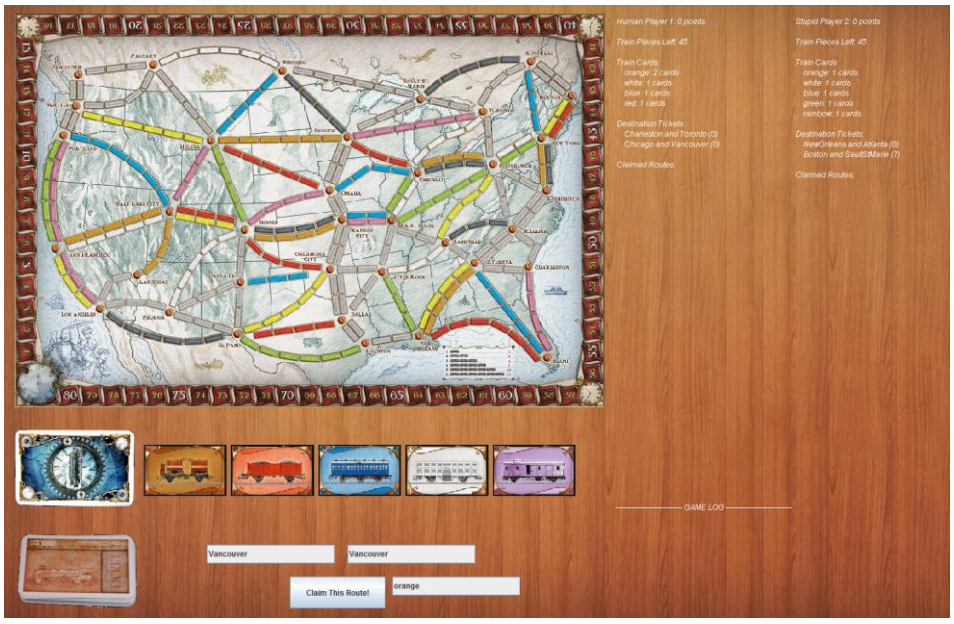
Addendum to Rules:

To (somewhat) simplify this assignment, we will be making a few adaptations to the rules of the game:

- *A player may only select one train card per turn (face up or from the pile) as opposed to two.*
- *When selecting more destination tickets, the player is given two destination tickets and may not return either of them to the stack.*
- *No points are awarded for the longest contiguous route.*

Background – The Simulator:

The simulator will be given to you via Collab. A screenshot of the game is shown below:



The most important class within the simulator is Player.java, an abstract class defining methods each of the (max of two) players can invoke. A summary of this class and its most important methods are shown in the table below:

Player.java

Player(String name)	*Constructs a new player. The parameter name is simply an identifying string used on the scoreboard to refer to this player.*
abstract void makeMove()	*Automatically invoked every time it is this player's turn. This method should call one of the three methods below to invoke the desired move for this turn.*
void drawTrainCard(int index)	*Invoke this method from your makeMove() method if you wish to draw a train card on this turn. An index of 0 indicates you'd like to pull a card from the top of the deck while a value of 1-5 indicates you'd like one of the face up cards.*
drawDestinationTickets()	*Adds two new destination tickets to your hand and ends your turn. Invoke from makeMove() if you'd like to draw more destination tickets this turn.*
claimRoute(Route route, TrainCardColor colorToUse)	*Invoke this method if you wish to purchase a route on the board that is unclaimed. You must provide a Route object (the route you wish to claim) along with the color card you wish to use as payment (e.g., some routes can be purchased with any color card). The simulator will automatically use rainbow cards if absolutely necessary to make the transaction work.*

You must create your own class that inherits Player.java and override / implement the makeMove() method. Your makeMove() may ONLY invoke the three methods described above. You may not change your player's state directly in any other way. The simulator is currently not airtight in this regard so we will be checking your code manually.

Getting Started:

First, extend the Player class and implement the makeMove() method. An example appears below:

```java
package ttr.model.player;

/**
 * A very stupid player that simply draws train cards only. Shown as an example of implemented a player.
 * */
public class StupidPlayer extends Player{

    /**
     * Need to have this constructor so the player has a name, you can use no parameters and pass the name of your player
     * to the super constructor, or just take in the name as a parameter. Both options are shown here.
     * */
    public StupidPlayer(String name) {
        super(name);
    }
    public StupidPlayer(){
        super("Stupid Player");
    }

    /**
     * MUST override the makeMove() method and implement it.
     * */
    @Override
    public void makeMove(){
        /* Always draw train cards (0 means we are drawing from the pile, not from the face-up cards) */
        super.drawTrainCard(0);
    }

}
```

To test your code, find the file ***TTRMain.java***. There are two lines in the middle that initialize the two players. Simply change one of these lines so that it instantiates your class instead of mine. If you want to play against your AI, then make sure one of the two players is a HumanPlayer. Likewise, if you'd like your AI to play against itself, have both players be instantiations of your class:

```java
package ttr.main;

import ttr.model.destinationCards.Routes;

public class TTRMain {

    public static void main(String[] args) {

        /* This is the game object required by the engine (essentially just the game window) */
        TicketToRide myGame = new TicketToRide();

        /* Initialize two players */
        Player player1 = new HumanPlayer("Human Player 1");
        Player player2 = new StupidPlayer("Stupid Player 2");

        /* Initialize the routes */
        Routes.initialise();

        TTRGamePlayScene scene = new TTRGamePlayScene("Ticket To Ride", "woodBacking.jpg", myGame, player1, player2);
        myGame.setCurrentScene(scene);
        myGame.start();
        scene.playGame();
    }
}
```

Simply run the code and watch the AI play.

Write-up:

Produce a document that describes, at a minimum, the following aspects of the assignment:

- Describe the algorithm you implemented. Why does it make sense to play the game this way? Is this the way a human would play, or is the strategy fundamentally different? Did you need to use heuristics, if so what did you choose and why?

- Describe in detail how you tested your strategy. How did you detect issues with your AI's approach? Give me some specific examples of testing you did and what these tests helped you learn about your approach to the game.

- Give me some data on how well your AI plays against some baselines. Maybe implement a quick random AI and see how well your implementation does. Analyze your results and discuss. If you tested against others, give me some data on how well your AI did. Where does your code fail and/or succeed? Why is that the case?

You Will Turn In:

…Please submit two files separately on Collab. One will be your inherited player class (just the one .java file only). The other is a single pdf of your write-up. If you do not submit your homework as specified you will lose points and be left out of the class competition.