8.4 Appendix 4: Source Code

Board.java

```
/**
 * @version 1.0
 * @author Tobias
 */

package diceGame;

import desktop_fields.*;
import desktop_resources.*;
import java.awt.Color;

public class Board {
	Field[] fields = new Field[21];
	Street[] graphicfields = new Street[fields.length];

	public Board(){
		//We can instantiate an object in the field array by using a subclass' constructor
		//Remember to use the appropriate constructors

		fields[0] = new Refuge(5000);					//Walled city
		fields[1] = new Tax(2000, -1);					//Goldmine (has -1 as tax rate so
we know it does not actually have a tax rate)
		fields[2] = new Territory(1000, 100);			//Tribe Encampment
		fields[3] = new Fleet(4000);					//Second sail
		fields[4] = new Territory(1500, 200);			//Crater
		fields[5] = new Territory(2000, 500);			//Mountain
		fields[6] = new LaborCamp(2500, 100);			//Huts in the mountain
		fields[7] = new Territory(3000, 700);			//Cold Desert
		fields[8] = new Fleet(4000);					//Sea Grover
		fields[9] = new Territory(4000, 1000);			//Black cave
		fields[10] = new Territory(4300, 1000);		//The WereWall
		fields[11] = new Refuge(500);					//Monastery
		fields[12] = new Tax(4000, 10);				//Caravan
		fields[13] = new Fleet(4000);					//The Buccaneers
		fields[14] = new LaborCamp(2500, 100);			//The Pit
		fields[15] = new Territory(4750, 1600);		//Mountain village
		fields[16] = new Territory(5000, 2000);		//South Citadel
		fields[17] = new Territory(5500, 2600);		//Palace Gates
		fields[18] = new Fleet(4000);					//Private Armada
		fields[19] = new Territory(6000, 3200);		//Tower
		fields[20] = new Territory(8000, 4000);		//Castle

		/*
		//Use messages class to set the names of the field class
		//String[] names = Messages.getFNames();			//We get the names
		for(int i = 0; i < fields.length; i++){
			fields[i].setName(Messages.getFNames()[i]);			//We set the names
		}
		*/

	}

	public Field[] getFields(){
		return fields;
	}
```

```
	//TODO Add to design diagrams
	public void showFieldsOnGUI(){
		//Show fields on GUI is a public method that can be used to make
		//the graphical representation of the fields with the GUI.
		//Board now creates the GUI board as well
		for(int i = 0; i < fields.length; i++){
			graphicfields[i] = new Street.Builder()

.setBgColor(determineFieldColor(i))

.setTitle(Messages.getFNames()[i])

.setDescription(Messages.getFNames()[i])

.setSubText(determineSubText(i))

.setRent(determineRent(i))
                                                                                                .build();
		}

		GUI.create(graphicfields);
		GUI.displayChanceCard();
	}

	private Color determineFieldColor(int fn){
		Color color;
		int red = 200;
		int green = 200;
		int blue = 200;

		int temp;
		double percentage;

		//We use instanceof to determine type of field
		if(fields[fn] instanceof Territory){
			//Territory should be green and varying in darkness after how expensive it is
			temp = ((Territory) fields[fn]).getRent();
			percentage = 100*(temp/4000.0);
			//System.out.print(percentage);

			red -= 1.9*percentage+10;
			green -= 1.4*percentage-40;
			blue -= 0.5*percentage+150;

		}
		else if(fields[fn] instanceof Fleet){
			//Fleet is blue
			red -= 100;
			green -= 100;
			blue += 40;
		}
		else if(fields[fn] instanceof LaborCamp){
			//LaborCamp is grey
			red -= 50;
			green -= 50;
			blue -= 50;
		}
		else if(fields[fn] instanceof Tax){
			//Tax is red
			red += 40;
			green -= 150;
```

```
                        blue -= 150;

                }
                else if(fields[fn] instanceof Refuge){
                        //Refuge is gold/yellow
                        red += 52;
                        green += 52;
                        blue += 52;
                }
                //color is assigned values of RGB
                color = new Color(red, green, blue);
                return color;
        }

        private String determineSubText(int fn){
                String text = "";
                if(fields[fn] instanceof Ownable){
                        text += Messages.getBMessages()[0]; //Price:
                        text += " " + String.valueOf(((Ownable) fields[fn]).getPrice());

                }
                else if(fields[fn] instanceof Tax){
                        text += Messages.getBMessages()[3]; //Pay:
                        text += " " + String.valueOf(((Tax) fields[fn]).getTaxAmount());
                        if(((Tax) fields[fn]).getTaxRate() >= 0){
                                text += " " + Messages.getBMessages()[4];
                                text += " " + String.valueOf(((Tax) fields[fn]).getTaxRate());
                                text += "% " + Messages.getBMessages()[5];
                        }
                }
                else if(fields[fn] instanceof Refuge){
                        text += Messages.getBMessages()[2]; //Recieve:
                        text += " " + String.valueOf(((Refuge) fields[fn]).getBonus());
                }
                return text;
        }
        // The GUI is not able to show messages on multiple lines
        // on the centerpiece of the GUI. The messages below have
        // been adjusted to fit with the GUI, so it might not be
        // obvious how "fleet" and "labor camp" works.
        private String determineRent(int fn){ //Rent:
                String rent = "";
                if(fields[fn] instanceof Territory){
                        rent += Messages.getGMessages()[22] + String.valueOf((((Territory) fields[fn]).getRent()));
                }
                else if(fields[fn] instanceof Fleet){
                        rent += Messages.getGMessages()[23];
                }
                else if(fields[fn] instanceof LaborCamp){
                        rent += Messages.getGMessages()[24];
                }

                return rent;
        }
}
```

CreateGame.java

```
/* This class can add the player amount and creates the game.
 * Currently this needs to be re-evaluated as it creates a game of 2, and then adds a game.! */

package diceGame;

import desktop_resources.GUI;

public class CreateGame {

	private int playerAmount;	//Number of players in the game
	private Game game;

	public CreateGame(){
		game = new Game(); //Initialize game
		while (GUI.getUserButtonPressed(Messages.getGMessages()[6] //do you want to create new game?
					, Messages.getGMessages()[1] //yes
					, Messages.getGMessages()[2] //no
					) == Messages.getGMessages()[1] //user chooses yes
				){
				playerAmount = Integer.parseInt(GUI.getUserSelection(Messages.getGMessages()[8],
"2","3","4","5","6"));
				//Maybe possibility to name players
				game.resetGame(playerAmount, 30000); //Reset the game with new amount of players and set
start balance
				game.playGame();
		}
		GUI.close();
	}
}
```

DiceCup.java

```
package diceGame;

public class DiceCup {
	protected int[] values;
	protected int sides;

//Constructor to set amount of dice-sides and the amount of dices
//furthermore uses the setAllValuesRandom method which simulates a roll
	public DiceCup(int diceSides, int diceAmount){
		values = new int[diceAmount];
		this.sides = diceSides;
		this.setAllValuesRandom();
	}

	public int[] getValues(){
		return values;
	}

	public int getSum(){
		int sum = 0;
		for (int i = 0; i<values.length;i++){
			sum += values[i];
		}
		return sum;
	}

	//Simulates a roll of the chosen dice(s)
	public void setAllValuesRandom(){
		for (int i = 0; i < values.length; i++){
			values[i] = ( (int) (Math.random()*sides)+1);
		}
	}
}
```

Field.java

```
/**
 * @version 1.0
 * @author Tobias
 */

package diceGame;

public abstract class Field {
	//private String name;

	/*
	 * Field is an abstract class.
	 * Having an abstract method means that every class that
	 * extends it will also have that method.
	 * We can then do different implementations of this method
	 * for each of our specific classes.
	 */
	public abstract void landOnField(Player player);

	/*
	public void setName(String text){
		name = text;
	}

	public String getName(){
		return name;
	}
	*/
}
```

Fleet.java

```
/**
 * @version 1.0
 * @author freya
 */

package diceGame;

public class Fleet extends Ownable {
	private final int RENT_1 = 500;
	private final int RENT_2 = 1000;
	private final int RENT_3 = 2000;
	private final int RENT_4 = 4000;

	public Fleet(int price) {
		super(price);
	}

	@Override
	public int getRent() {
		int ownedFleets = 0;

		for (int i = 0; i<owner.getOwnedFields().length; i++){
			if (owner.getOwnedFields()[i] instanceof Fleet){
				ownedFleets++;
			}
			else if (owner.getOwnedFields()[i] == null){
				break;
			}
		}

		int rent = 0;

		switch (ownedFleets){
		case 1: rent = RENT_1; break;
		case 2: rent = RENT_2; break;
		case 3: rent = RENT_3; break;
		case 4: rent = RENT_4; break;
		default: rent = 0; break;
		}

		return rent;
	}
}
```

Game.java

```
package diceGame;

import java.awt.Color;

import desktop_codebehind.Car;
import desktop_resources.GUI;

public class Game {
	protected Board board; //An instance of the Board class
	protected Player[] players; //An array of Players
	protected DiceCup dice; //An instance of the DiceCup class
	protected final int diceAmount; //The amount of dice used by the game
	protected final int diceSides; //The number of sides the dice can have
	protected Field currentField;

	public Game(){
		diceAmount = 2;			diceSides = 6;
		dice = new DiceCup(diceSides,diceAmount);
		board = new Board();
		board.showFieldsOnGUI();
	}

	public void resetGame(int playerAmount, int balance){
		int startBalance = balance; //This cannot be a final as we need to be able to reset the game from the GUI
		players = new Player[playerAmount];
		Color color = null;
		for (int i = 0; i < players.length; i++){
			switch (i){
			case 0: color = Color.red; break;
			case 1: color = Color.green; break;
			case 2: color = Color.yellow; break;
			case 3: color = Color.blue; break;
			case 4: color = Color.white; break;
			case 5: color = Color.black; break;
			default: System.exit(1);
			}
			players[i] = new Player(Messages.getGMessages()[10]+(i+1),i+1,startBalance, new
Piece(color));
			Car car = new Car.Builder()
					.primaryColor(players[i].getPiece().getColor())
					.build();
			GUI.addPlayer(players[i].getName(), players[i].getBalance(), car);
			GUI.setBalance(players[i].getName(), players[i].getBalance());
			GUI.removeAllCars(players[i].getName());
		}
		for (int i = 0; i<board.getFields().length; i++){
			GUI.removeOwner(i+1);
			if (board.getFields()[i] instanceof Ownable && ((Ownable) board.getFields()[i]).getOwner() !=
null){
				((Ownable) board.getFields()[i]).setOwner(null);
			}
		}
	}

	public void playGame(){
		boolean winnerFound = false;
		Player currentPlayer;
```

```
		//first player is player 1
		currentPlayer = players[0];

		while (winnerFound == false){
			currentPlayer = playTurn(currentPlayer);

			if (players.length == 1){
				winnerFound = true;
			}
		}

		GUI.showMessage(Messages.getGMessages()[14] + currentPlayer.getName() +
Messages.getGMessages()[15]);

	}

	protected Player playTurn(Player currentPlayer){
		GUI.getUserButtonPressed(Messages.getGMessages()[11] + currentPlayer.getName() +
Messages.getGMessages()[12], Messages.getGMessages()[7]);

		throwDice(currentPlayer);

		movePiece(currentPlayer);

		currentField.landOnField(currentPlayer);

		Player nextPlayer = defineNextPlayer(currentPlayer);
		return nextPlayer;
	}

	protected void removePlayer(Player player){
		Player[] temp;
		temp = players;

		players = new Player[temp.length-1];

		int playerCount = 0;
		for (int i = 0; i<temp.length;i++){
			if (temp[i] != player){
				players[playerCount] = temp[i];
				playerCount++;
			}
		}

		GUI.removeAllCars(player.getName());
	}

	protected Player defineNextPlayer(Player currentPlayer){
		Player nextPlayer;

		if (currentPlayer == players[players.length-1]){
			nextPlayer = players[0];
		}
		else{
			//find currentPlayer's index in players
			int arrayIndex = 0;
			for (int i=0;i<players.length;i++){
				if (currentPlayer == players[i]){
					arrayIndex=i;
				}
			}
			nextPlayer = players[arrayIndex+1];
```

```
                }

                //remove player if balance = 0
                if (currentPlayer.getBalance() == 0){
                        removePlayer(currentPlayer);
                }

                if (players.length == 1){
                        nextPlayer = players[0];
                }

                return nextPlayer;
        }

        protected void throwDice(Player currentPlayer){
                dice.setAllValuesRandom();
                currentPlayer.setDiceSum(dice.getSum());
                GUI.setDice(dice.getValues()[0], dice.getValues()[1]);
        }

        protected void movePiece(Player currentPlayer) {
                if (currentPlayer.getPiece().getPosition() != 0){
                        /*
                         * If there has already been placed a car, we remove it before placing a new one
                         * We avoid bugs in the first turn by having the position set to 0
                         * Every field is then a value of 1-21.
                         */
                        GUI.removeCar(currentPlayer.getPiece().getPosition(),currentPlayer.getName());
                }
                /*
                 * Position is equal to the current position + the sum of the dice throw
                 * We use modulus to calculate whether the player would end up outside the board
                 */
                int position = (currentPlayer.getPiece().getPosition() + dice.getSum()) % board.getFields().length;
                //Since we use mod 21, we need to have a special case for field 21, or else we get position == 0
                if (position == 0){
                        position = board.getFields().length;
                }

                //We set the car and piece position to the new values
                currentPlayer.getPiece().setPosition(position);
                GUI.setCar(position, currentPlayer.getName());
                currentField = board.getFields()[position-1];
                GUI.displayChanceCard(Messages.getFNames()[position-1] + "<br><br>" +
Messages.getFMessages()[position-1]);

        }

}
```

LaborCamp.java

```
/**
 * @version 1.1
 * @author freya
 */

package diceGame;

public class LaborCamp extends Ownable {
	private int baseRent;

	public LaborCamp(int price, int baseRent) {
		super(price);
		this.baseRent = baseRent;
	}

	@Override
	public int getRent() {
		int ownedLaborCamps = 0;

		for (int i = 0; i<owner.getOwnedFields().length;i++){
			if (owner.getOwnedFields()[i] instanceof LaborCamp){
				ownedLaborCamps++;
			}
			else if (owner.getOwnedFields()[i] == null){
				break;
			}
		}

		return baseRent*ownedLaborCamps;
	}
}
```

Messages.java

```
package diceGame;

public class Messages {
	private static String[] fieldMessages = {
/*Field 1*/			"Du ankommer til en fæstning med meget høje mure. Indbyggerne kan ikke komme ud. De betaler dig 5000 som tak, da du foreslår dem at lave en port i muren."
/*Field 2*/			,"Du ankommer til en Guldmine! En masse sure dværge kommer ud og stjæler 2000 fra dig."

/*Field 3*/			,"Du ankommer til en Lejr hvor alle beboerne stammer."

/*Field 4*/			,"Du ankommer til en havn hvor det berømte skib 'Sejl nr. 2' ligger til kaj."

/*Field 5*/			,"Du ankommer til et krater."

/*Field 6*/			,"Du ankommer til et bjerg."

/*Field 7*/			,"Du ankommer til nogle bjerghytter der er beboet af venlige nisser."

/*Field 8*/			,"Du ankommer til en gold og forfrossen ørken."

/*Field 9*/			,"Du ankommer til en havn hvor det gigantiske lyserøde skib 'Hav Crover' ligger til kaj."

/*Field 10*/			,"Du ankommer til en meget uhyggelig mørk grotte der er til salg."
/*Field 11*/			,"Du ankommer til den berygtede varulvemur."
/*Field 12*/			,"Du ankommer til et kloster. Her tilbeder munkene pengeguden Yll'an. De giver dig 500."

/*Field 13*/			,"Du ankommer til en campingvogn. Ud kommer der en hillbilly med et haglgevær. Betal ham 4000 eller 10% af alt hvad du ejer."
/*Field 14*/			,"Du møder en venlig pirat, der giver dig en enestående mulighed for at købe hans sørøverskib."

/*Field 15*/			,"Du ankommer til et stort hul i jorden."
/*Field 16*/			,"Du ankommer til en bjergby."
/*Field 17*/			,"Du ankommer til Den sydlige hovedstad. Borgmesteren er virkelig fuld, og han tilbyder dig at købe hele byen."
/*Field 18*/			,"Du ankommer til et meget stort og flot palads."
/*Field 19*/			,"Du møder kaptajnen for en flåde af lejesoldater."
/*Field 20*/			,"Du møder en gal troldmand i et højt tårn. Han tryller sine egne bukser om til guld, og tilbyder dig at købe tårnet."
/*Field 21*/			,"Du ankommer til kongerigets slot."
		};
		private static String[] fieldNames = {
					"Fæstning"				//Field 1
					,"Guldmine"				//Field 2
					,"Stamme Lejr"			//Field 3
					,"Sejl nr. 2"			//Field 4
					,"Krater"				//Field 5
					,"Bjerg"				//Field 6
					,"Bjerghytter"			//Field 7
					,"Kold Ørken"			//Field 8
```

```
                        ,"Hav Grover"                                   //Field 9
                        ,"Grotte"                                       //Field 10
                        ,"Varulvemuren"                                 //Field 11
                        ,"Kloster"                                      //Field 12
                        ,"Campingvogn"                                  //Field 13
                        ,"Sørøverskibet"                        //Field 14
                        ,"Hullet"                                       //Field 15
                        ,"Bjergby"                                      //Field 16
                        ,"Den sydlige hovedstad"        //Field 17
                        ,"Palads"                                       //Field 18
                        ,"Lejesoldater"                                 //Field 19
                        ,"Tårn"                                                 //Field 20
                        ,"Slot"                                                 //Field 21
        };
        private static String[] boardMessages = {
                        "Pris:",                                //0
                        "Leje:",                                //1
                        "Modtag:",                                      //2
                        "Betal:",                               //3
                        "eller",                                //4
                        "af alle ejendele",             //5
        };

        private static String[] generalMessages = {
        /*0*/       "Denne ejendom er ikke ejet af nogen spiller. Vil du købe den for ",
        /*1*/       "Ja",
        /*2*/       "Nej",
        /*3*/       "Du har nu to muligheder",
        /*4*/       "Betal ",
        /*5*/       "% af alle ejendele ",
        /*6*/       "Vil du starte et nyt spil?",
        /*7*/       "Slå med terningerne",
        /*8*/       "Hvor mange spillere skal deltage i spillet?",
        /*9*/       "Du er landet på en anden spillers ejendom. Du skal betale ",
        /*10*/      "Spiller ",
        /*11*/      "Det er ",
        /*12*/      "'s tur.",
        /*13*/      "Du er landet på en arbejdslejr og skal slå med terningerne, for at bestemme hvor meget du skal betale i
leje.",
        /*14*/      "Tillykke ",
        /*15*/      ", du har vundet spillet!",
        /*16*/      " i leje.",
        /*17*/      "Du skal betale ",
        /*18*/      " til skattefar.",
        /*19*/      "Du modtager ",
        /*20*/      "Du er landet på din egen ejendom og nyder de dejlige omgivelser.",
        /*21*/      "Ejeren af denne ejendom er gået bankerot, og du slipper derfor for at betale leje.",
        /*22*/      "Leje: ",
        /*23*/      "Leje: 500, 1000, 2000, 5000",
        /*24*/      "Leje: 100*øjne*labor camps ejet",
        /*25*/  "Du har ikke nok penge til at købe dette felt.",
        };

        public static String[] getFMessages(){
                return fieldMessages;
        }

        public static String[] getFNames(){
                return fieldNames;
        }

        public static String[] getBMessages(){
```

```
		return boardMessages;
	}

	public static String[] getGMessages(){
		return generalMessages;
	}

}
```

Ownable.java

```
/**
 * @version 1.2
 * @author freya, tobias
 */

package diceGame;

import desktop_resources.GUI;

public abstract class Ownable extends Field {
	protected int price;
	protected Player owner;

	public Ownable(int price){
		this.price = price;
		this.owner = null;
	}

	public abstract int getRent();

	public Player getOwner(){
		return owner;
	}

	public int getPrice(){
		return price;
	}

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

	public void landOnField(Player player){
		if (owner == null && player.getBalance() >= price){
			if (GUI.getUserButtonPressed(
					player.getName() + ": " + Messages.getGMessages()[0] + price + "?" //Do
you want to buy field?
					,Messages.getGMessages()[1] //Yes
					,Messages.getGMessages()[2] //No
					) == Messages.getGMessages()[1])
			{//user chooses yes
				owner = player;
				player.setOwnedField(this);
				player.setBalance(player.getBalance()-price);
				GUI.setOwner(player.getPiece().getPosition(), player.getName());
			}
		}
		else if(owner == null && player.getBalance() < price){
			GUI.showMessage(player.getName() + ": " + Messages.getGMessages()[25]);
		}
		else if (owner == player){
			GUI.showMessage(player.getName() + ": " + Messages.getGMessages()[20]);
		}
		else if (owner.getBalance() > 0){//pay rent to owner if he is not bankrupt
			int rent = 0;
			if (this instanceof LaborCamp){
				//when LaborCamp we should multiply dice sum with 100 and number of owned labor
camps
```

```
                                        rent = getRent()*player.getDiceSum();
                                }
                                else{
                                        rent = getRent();
                                }

                                GUI.showMessage(player.getName() + ": " + Messages.getGMessages()[9] + rent +
Messages.getGMessages()[16]);

                                owner.setBalance(owner.getBalance() + rent);
                                player.setBalance(player.getBalance() - rent);
                        }
                        else{
                                GUI.showMessage(player.getName() + ": " + Messages.getGMessages()[21]);
                        }

                }
}
```

Piece.java

```
/* This class has the responsibility of handling the player's piece/car.
 * It carries two private variables:
 *                      - An integer with the current board-position of the piece
 *                      - A color for the piece's color.
 */

package diceGame;
import java.awt.Color;


public class Piece {
        private int position;
        private Color color;

        // Constructor - In order to create the piece, you will need to give the vehicle a color using java.awt.Color;
        public Piece(Color color) {
                this.color = color;
                position = 0;
        }

        public Color getColor() {
                return this.color;
        }

        // Move the piece to a position using an integer
        public void setPosition(int position) {
                this.position = position;
        }

        public int getPosition() {
                return this.position;
        }

}
```

Player.java

```
package diceGame;

import desktop_resources.GUI;

public class Player {

	private int id;
	private int balance;
	private String name;
	private int diceSum;
	private Piece piece;
	private Ownable[] ownedFields;

	public Player(String name,int id, int balance, Piece piece){
		this.name = name;
		this.id = id;
		this.balance = balance;
		this.piece = piece;
		ownedFields = new Ownable[17];
		diceSum = 0;

	}

	public String getName(){

		return name;

	}

	public int getBalance(){

		return balance;

	}

	public Piece getPiece(){

		return piece;

	}

	public int getID(){

		return id;

	}

	public Ownable[] getOwnedFields(){

		return ownedFields;

	}

	public int getDiceSum(){
		return diceSum;
	}

	public void setBalance(int balance){
```

```
		//we shall set balance to 0 if it is negative
		if (balance < 0){
			balance = 0;
		}
		this.balance = balance;

		GUI.setBalance(name, balance);
	}

	public void setOwnedField(Ownable field){
		//set next empty position in Ownable array to the given field
		for (int i = 0; i<ownedFields.length;i++){
			if (ownedFields[i] == null){
				ownedFields[i] = field;
				break; //exit the loop
			}
		}

	}

	public void setDiceSum(int diceSum){
		this.diceSum = diceSum;
	}

}
```

Refuge.java

```
package diceGame;

import desktop_resources.GUI;

public class Refuge extends Field {
	private int bonus;

	public Refuge (int bonus){
		this.bonus = bonus;
	}

	public int getBonus(){
		return bonus;
	}

	public void landOnField(Player player){
		GUI.showMessage(player.getName() + ": " + Messages.getGMessages()[19] + bonus + ".");
		player.setBalance(player.getBalance() + bonus);
	}
}
```

StartProgram.java

```
package diceGame;

public class StartProgram {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		//Instantiates the CreateGame method which is the only
		//responsibility for this class
		new CreateGame();

	}

}
```

Tax.java

```
/**
 * @version 1.0
 * @author freya
 */
package diceGame;

import desktop_resources.GUI;

public class Tax extends Field {
	private int taxAmount;
	private int taxRate; //taxRate = -1 means no taxRate.

	public Tax(int taxAmount, int taxRate){
		this.taxAmount = taxAmount;
		this.taxRate = taxRate;
	}

	public int getTaxAmount(){
		return taxAmount;
	}

	public int getTaxRate(){
		return taxRate;
	}

	public void landOnField(Player player){
		if (taxRate >= 0){
			String response = GUI.getUserButtonPressed(player.getName() + ": " +
			Messages.getGMessages()[3]				//You have two options
			,Messages.getGMessages()[4] + taxAmount //Pay taxAmount
			,Messages.getGMessages()[4] + taxRate + Messages.getGMessages()[5] //Pay taxRate
								);
			if (response.equals(Messages.getGMessages()[4] + taxAmount)){//user chooses taxAmount
				player.setBalance(player.getBalance() - taxAmount);
			}
			else{//user chooses taxRate
				player.setBalance(player.getBalance() - (int)((taxRate/100.0) * getAllAssets(player)));
			}
		}
		else{
			GUI.showMessage(player.getName() + ": " + Messages.getGMessages()[17] + taxAmount +
Messages.getGMessages()[18]);
			player.setBalance(player.getBalance() - taxAmount);
		}
	}
	private int getAllAssets(Player player){
		Ownable[] fields = player.getOwnedFields();
		int ownedAssets = player.getBalance();

		for (int i = 0; i < fields.length; i++){
			if (fields[i] != null){
				ownedAssets += fields[i].getPrice();
			}
		}

		return ownedAssets;
	}
}
```

Territory.java

```
/**
 * @version 1.0
 * @author freya
 */

package diceGame;

public class Territory extends Ownable {
	private int rent;

	public Territory(int price, int rent) {
		super(price);
		this.rent = rent;
	}

	@Override
	public int getRent() {
		return rent;
	}
}
```