

Mobile App Design Project #1

Java Boot Camp: Design Model for Chutes and Ladders Board Game

Directions:

In mobile Applications – the “Control – Model – View” model works to divide the work within an application. The “Control” talks in user and sensor data and passes it to the “Model.” The “Model” uses data structures, Classes, instances, algorithms, and functions to calculate the Game state at specific intervals. The “Model” then passes this data to the “View” which draws our outputs Game data to the screen for the user/players.

In this lesson, we will develop the Model for a basic board game. There are two purposes to this activity:

1. Develop a common vocabulary of Java Data types and coding techniques that we will use in this class. For beginners – this is the time to learn enough Java to “be dangerous.” For advanced Java programmers – this introduces the technique of creating code and classes from ‘scratch’ to build Game logic.
2. Build the “Model” aspect to a Game program –develop the Classes and algorithms to run game Data. We will not use a graphic display for this project. We will use Eclipse and the Console (Text) output.

Chutes and Ladders Class Model:

What do we need to play Chutes and Ladders?



We need:

Players

Spinner

Game Board

In developing a Game Program – the designer will first sketch out the Classes and Data types for the Game. For Chutes and Ladders we will have the following Classes: (Note the use of Pseudo Code)

Class Player:

Fields:

Private String name: Stores the Name of the Player instance

Private Int position: Stores the Players position on the Game Board

Constructor:

Takes the argument to set the Player's name

Sets the position to 0

Modifier Functions (I call these "Setters")

Public void setPosition(int input)

Public void setName(String n)

Public void changePosition(int input) -> This will be used after the Player "spins"

Access Functions (I call these "Getters")

Public int getPosition() -> Returns players current Position

Public String getName() -> Returns the players name

Other Functions

Public void spin(Spinner s) -> the player "spins" the spinner and moves to a new position.

Class Spinner: (Used by the players)

Import java.util.Random -> For Random numbers

Fields:

Private int number -> range of spinner

Constructor:

Takes the input of a number integer and sets it to 'number'

Functions:

Public int getSpin() ->

Creates a Random object called 'generator'

Gets a number from 'generator'

Returns the number

Class Display (This will play the role of the Game Board and the Game Logic)

Fields (All Static and used within the main function)

Array Player[] playerList

Array ladders[][] (9 by 2 array)

Array chutes[][] (10 by 2 array)

Functions (Static)

Void checkLadder(Player p) – checks if Player p landed on a Ladder

Void checkLadder(Player p) – checks if Player P landed on a chute

Game Logic (also the display and the main method)

1. Build the Ladders
2. Build the Chutes
3. Create Instance of Spinner with 6 positions
4. Create Four players
 - a. "Joshua"
 - b. "Daddy"
 - c. "Rebecca"
 - d. "Mommy"
5. Store these players in the playerList
6. Game Logic:
 - a. The players take turns:
 - i. Spin
 - ii. Check Ladders
 - iii. Check Chutes
 - iv. Display the Player Position
 - v. If Player Position is > 100 – end Game and Display the Winner

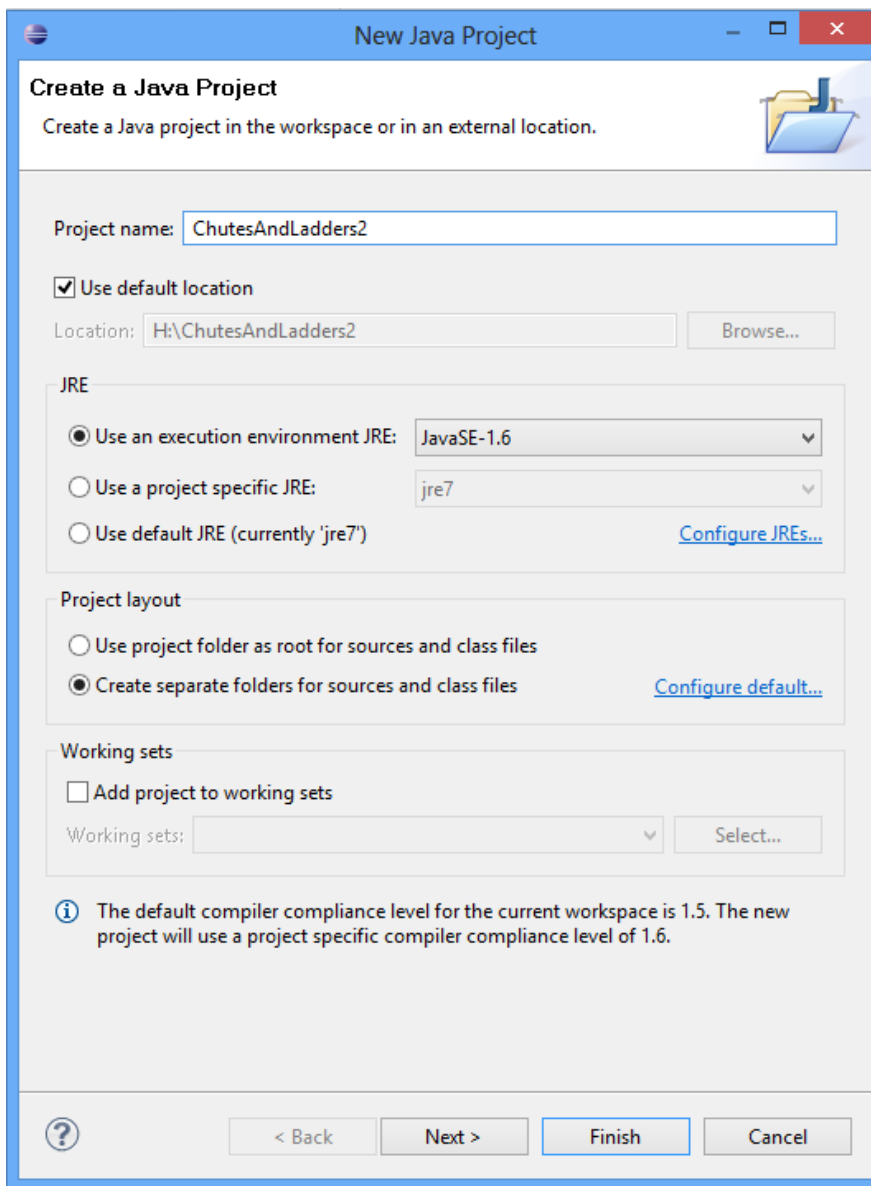
*** Please note – Can we accomplish this program with One Class and one main method? Yes, of course. However, when designing Apps and Games – the programmer must work to divide the program into components that can be modified and reused. That is the strength of an Object Orientated

program like Java. As we develop more complex programs – the creating of classes and modules will greatly empower the development process.***

Directions:

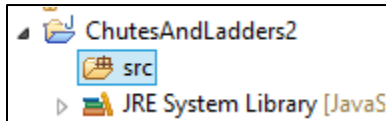
Phase 1: Create the Java Project for Chutes and Ladders

1. Start Eclipse and select New -> Java Project
2. Name the Project "ChutesAndLadders." (Note for my example I will use "ChutesAndLadders2" as I already have a project named ChutesAndLadders in my library.)
3. Set the Location to your SkyDrive JavaProjects Folder. (Or to where you store your Java projects)
4. Click Next and Finish

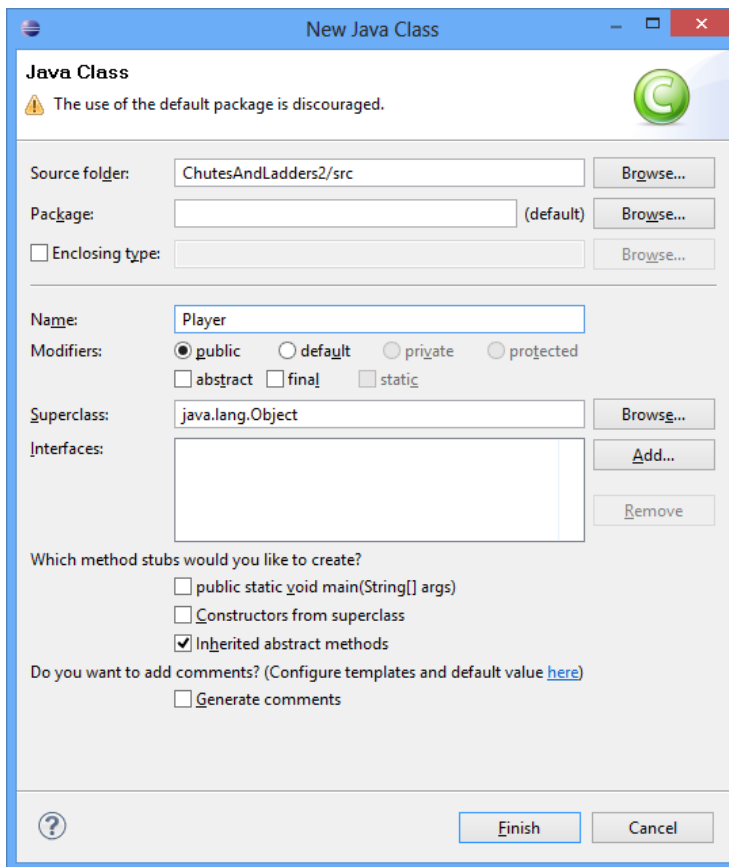


Phase 2: Create the Three classes (“Player”, “Spinner”, and “Display”)

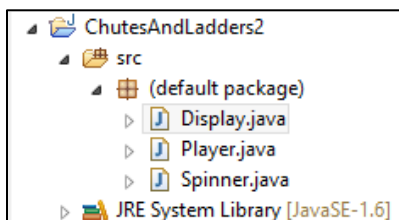
1. Select the Triangle to “Unfold” the ChutesAndLadders project. Click on the icon that says “src” (stands for source).



2. Right Click on src and select “New -> Class.”
3. Name the Class “Player” and click “Finish.”



4. Right Click on “src” again and select “New -> Class” and create the “Spinner” class.
5. Repeat the process to create a class called “Display.”
6. You will now have three classes in your src folder:



Phase 3: Write the “Player Class”

This class will model the data needed for our game players. Note the fields, constructor, and functions as described below:

```
Class Player:
  Fields:
    Private String name: Stores the Name of the Player instance
    Private int position: Stores the Players position on the Game Board

  Constructor:
    Takes the argument to set the Player's name
    Sets the position to 0

  Modifier Functions (I call these "Setters")
    Public void setPosition(int input)
    Public void setName(String n)
    Public void changePosition(int input) -> This will be used after the Player "spins"

  Access Functions (I call these "Getters")
    Public int getPosition() -> Returns players current Position
    Public String getName() -> Returns the players name

  Other Functions
    Public void spin(Spinner s) -> the player "spins" the spinner and moves to a new
    position.
```

1. Create the fields for name and position within the Player class code: (Lines 1 to 10)
 - a. Note the use of comments to describe the fields and code.
 - b. Sometimes as you are creating code – errors and warnings will pop up. Many of these warnings appear as we have not completely defined or used the classes and variables we have defined. As we complete the code, the errors will resolve.

```
1 // Player Class
2 // Represents Player in the Chutes and Ladders Game
3
4 public class Player {
5
6     // Fields for Player Class
7     private int position; // Stores position on Chutes and Ladders Board
8     private String name; // Stores a String with the Player instance name.
9
10 }
```

2. Code the Constructor for the Class: (Lines 10 to 14)

```
1 // Player Class
2 // Represents Player in the Chutes and Ladders Game
3
4 public class Player {
5
6     // Fields for Player Class
7     private int position; // Stores position on Chutes and Ladders Board
8     private String name; // Stores a String with the Player instance name.
9
10    // Constructor - takes a String to set the Player's name
11    public Player(String n){
12        position = 0; // Starting position is 0
13        name = n; // Sets name
14    }
15
16 }
17
```

3. Write the Code for the Modifiers (“Setters”) (Lines 16 to 30)

- a. Note: If you get errors with the Curly Brackets – make sure you have the current brackets for opening and closing functions. Often errors come from ‘extra’ brackets or ‘lack of’ brackets.

```
16 // Modifiers
17 // Set Position
18 public void setPosition(int input){
19     position = input;
20 }
21
22 // Set name (if it needs to be changed)
23 public void setName(String n){
24     name = n;
25 }
26
27 // Change Position - used by Display if landing on a chute or a ladder
28 public void changePosition(int input){
29     position = position + input;
30 }
```


4. Write the Code for the Access. (The “Getters”) (Lines 31 to 40)

```
31
32 // Access - get position and name Field data
33 public int getPosition(){
34     return position;
35 }
36
37 public String getName(){
38     return name;
39 }
40
```

5. Write the Function that gives the Player the ability to “Spin.” (Lines 40 to 47)
- a. Note the error in line 43. We have not written the Spinner Class function ‘getSpin()’ – so this will show as an error. This will correct itself after we write the Spinner Class.

```
40
41 // Function - Spin with a Spinner object and move to new position.
42 public void spin(Spinner s) {
43     int newSpin = s.getSpin();
44     System.out.println(name + "'s spin was: " + newSpin); // For Testing
45     position = position + newSpin;
46 }
47
```

Phase 4: Write the "Spinner" Class

Now we will create the Spinner Class. Recall the fields and functions:

```
Class Spinner: (Used by the players)
  Import java.util.Random -> For Random numbers
  Fields:
    Private int number -> range of spinner

  Constructor:
    Takes the input of a number integer and sets it to 'number'

  Functions:
    Public int getSpin() ->
      Creates a Random object called 'generator'
      Gets a number from 'generator'
      Returns the number
```

1. Select the "Spinner" class tab and enter the following code.

```
1 // Spinner Class
2 // Used by Player object to generate a random number
3
4 import java.util.Random;
5
6 public class Spinner {
7
8     // Field - An integer named 'number'
9     private int number;
10
11     // Constructor - takes in an integer representing the number of
12     // "Chances" on the Spinner
13     public Spinner(int n){
14         number = n;
15
16     }
17
18     // Function to return a Random Number - the "Spin"
19     public int getSpin(){
20         Random generator = new Random();
21         int newSpin = generator.nextInt(number) + 1;
22         return newSpin;
23     }
24
25 } // End Spinner Class
26
```

*** Note, check Player class again on line 43. If the 's.getSpin()' still shows as an error, retype the line and it should correct itself.***

Phase 5: Test the Player and Spinner Class with the “Display” class

We will use the “Display” class to run tests on our Player and Spinner Classes. We will run this test in several parts:

Part 1: Create an instance of Spinner and Player and then call the “Getter” Functions and display data to the Console.

Part 2: Use a For Loop to simulate one Player instance “spinning” and advancing to 100.

Part 3: Create Several Player Instances and use an Array to organize the “Taking Turns” with these Players.

Part 1: Instance of Spinner and Player:

1. Type the following Code into the Display Class to create an instance of Spinner and Player:

```
1
2 public class Display {
3
4     // Test the Player with an Instance
5     // Use the 'main' method to run tests
6     public static void main(String[] args) {
7
8         // Create a Player Instance
9         Player joshua = new Player("Joshua");
10
11        // Create a Spinner Instance
12        Spinner spinner = new Spinner(6);
13
14    }
15
16 }
17
```

*Note that ‘joshua’ is an instance of Player. ‘joshua’ has the name “Joshua” stored in the name field.

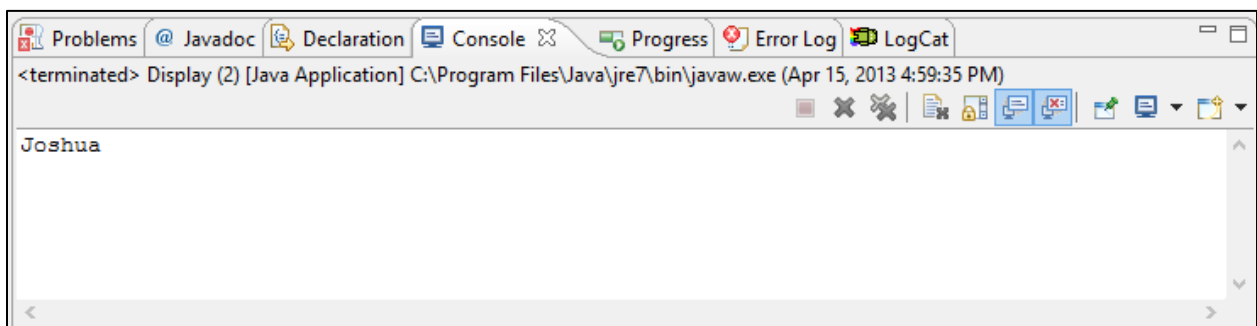
*Note that line 12 creates a new Spinner named ‘spinner’. ‘spinner’ can pick random numbers up to 6.

2. We will now test the 'joshua' instance. (I use the name 'joshua' because my son Joshua's favorite game was Chutes and Ladders – and I thought of this activity after hours and hours of playing this game with him. . . .)

Type this code and run the program. (Lines 14 and 15) The console should output "Joshua" -> the String name of the instance.

```
14 // Print out joshua's name
15 System.out.println(joshua.getName());
```

Console Output:

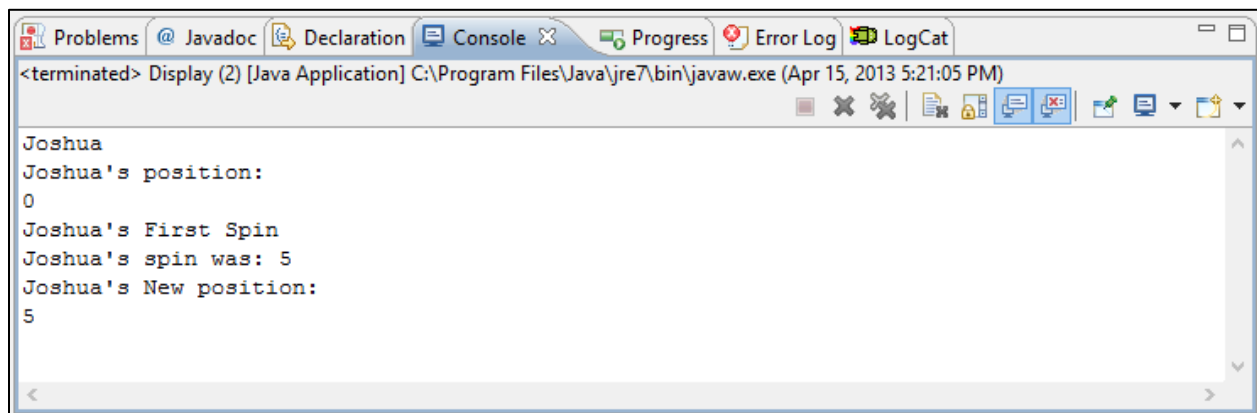


3. Now we will have Joshua "spin" and move to "new position." This will have three steps:
 - a. Print joshua's current Position
 - b. Call the joshua.spin function
 - c. Print the new Position for joshua.

 - Note that we will combine String and int data in the System.out.println() statement

```
16
17 // joshua Spins one time
18 System.out.println(joshua.getName() + "'s position:");
19 System.out.println(joshua.getPosition());
20 System.out.println(joshua.getName() + "'s First Spin");
21 // joshua spins
22 joshua.spin(spinner);
23 System.out.println(joshua.getName() + "'s New position:");
24 System.out.println(joshua.getPosition());
25
```

The Console should look like this:



The image shows a screenshot of an IDE's console window. The title bar includes tabs for 'Problems', 'Javadoc', 'Declaration', 'Console', 'Progress', 'Error Log', and 'LogCat'. The console text reads: '<terminated> Display (2) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Apr 15, 2013 5:21:05 PM)'. The output consists of the following lines: 'Joshua', 'Joshua's position:', '0', 'Joshua's First Spin', 'Joshua's spin was: 5', and 'Joshua's New position: 5'. The window has a standard toolbar with icons for file operations and a scroll bar on the right side.

```
<terminated> Display (2) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Apr 15, 2013 5:21:05 PM)
Joshua
Joshua's position:
0
Joshua's First Spin
Joshua's spin was: 5
Joshua's New position:
5
```

Part 2: Using a While Loop to have Joshua Spin until he reaches 100

We will now have joshua spin until his position is greater than 100. We will use a While Loop to create this model.

1. Type the following code: (Lines 28 to 35)

```
28
29     while (joshua.getPosition() < 100) {
30         System.out.println(joshua.getName() + "'s position: " + joshua.getPosition());
31         joshua.spin(spinner);
32     }
33
34     System.out.println(joshua.getName() + " has reached 100.");
35
```

Note the structure of the “while loop” on line 29:

`while (joshua.getPosition() < 100)` means that the code inside the loop will keep running until the Player joshua’s position is greater than 100.

Line 30 prints Player joshua’s current position.

Line 31 has Player joshua ‘spin’ the spinner and move to a new position.

Line 34 informs the user that Player joshua has reached 100.

Often Game Logic will use a ‘while loop’ to keep the game running until some type of condition or goal is met.

Part 3: Having Multiple Players Take Turns Spinning

We will now have 4 Player instances take turns Spinning the spinner until one of them reaches 100. The algorithm for this process will be:

- Create 4 Player instances
- Store these 4 Players in an Array
- Set a Boolean Variable to measure store the game state and control the While Loop to end the game
- While Loop to run the Game Logic
- Use a For Loop inside the While Loop for the Players to “take turns.”

1. Delete the void main function in the Display Class and set up the Display Class code to look like this:

```
1
2 public class Display {
3
4
5
6 }
7
```

2. Create the main Function and 4 instances of Players as shown below: (Lines 4 to 12)

```
1
2 public class Display {
3
4     // Test the Player with an Instance
5     // Use the 'main' method to run tests
6     public static void main(String[] args) {
7
8         // Create 4 Player Instances
9         Player joshua = new Player("Joshua");
10        Player rebecca = new Player("Rebecca");
11        Player daddy = new Player("Daddy");
12        Player mommy = new Player("Mommy");
13
```

3. Create an Array object with a length of 4. (Lines 13 to 16)

```
13
14     // Create an Array to hold the 4 Players
15     Player players[] = new Player[4];
16
```

4. Assign the Players to the players[] array positions: (Lines 17 to 21)

```
13
14     // Create an Array to hold the 4 Players
15     Player players[] = new Player[4];
16
17     // Assign the Players
18     players[0] = joshua;
19     players[1] = rebecca;
20     players[2] = daddy;
21     players[3] = mommy;
22
```

5. Create a Spinner instance: (Lines 23 and 24)

```
22
23     // Create a Spinner Instance
24     Spinner spinner = new Spinner(6);
25
```

6. Create a Boolean Variable to store the Game State. Set the variable to 'true.'

```
24
25     // Boolean Variable to Store Game State
26     boolean gameActive = true;
27     String winner = ""; // Stores name of winner
28
```

7. Establish the While Loop for the Game Logic.

```
28
29     // While Loop for the Game Logic
30     while (gameActive) {
31
32     }
33
```

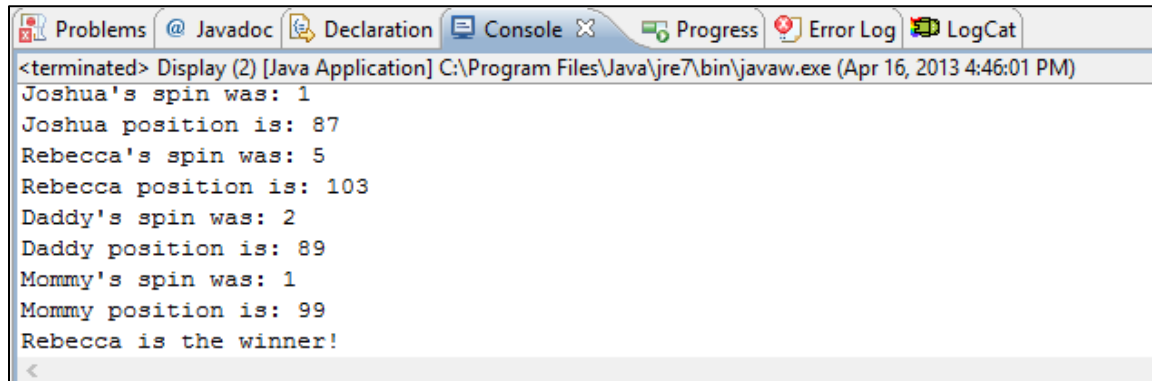

8. Inside the While Loop – set up a For Loop for the players to take turns.

```
32 // While Loop for the Game Logic
33 while (gameActive) {
34
35     // For Loop for Players to take turns
36     for (int p = 0; p < players.length; p++) {
37         players[p].spin(spinner);
38         System.out.println(players[p].getName() + " position is: " + players[p].getPosition());
39
40         // If the player reaches 100, end the While loop
41         if (players[p].getPosition() > 100) {
42             // Get the winner's name
43             winner = players[p].getName();
44             // Set the gameActive to false to exit the While loop
45             gameActive = false;
46         } // end if
47
48     } // end for
49
50 } // end while
51
```

9. Add a line of code after the while loop to display the winner:

```
51
52     System.out.println(winner + " is the winner!");
53
```

10. Save and Run the Program. The Console should look something like this:



```
<terminated> Display (2) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Apr 16, 2013 4:46:01 PM)
Joshua's spin was: 1
Joshua position is: 87
Rebecca's spin was: 5
Rebecca position is: 103
Daddy's spin was: 2
Daddy position is: 89
Mommy's spin was: 1
Mommy position is: 99
Rebecca is the winner!
<
```

Here is a problem – even if a player reaches 100, the other players still complete their spins because of the for loop in line 36. To fix this, we will add another set of if statements. Modify the while loop to add the if statement before the `players[p].spin(spinner)` command (Lines 38 to 42).

```
32     // While Loop for the Game Logic
33     while (gameActive) {
34
35         // For Loop for Players to take turns
36         for (int p = 0; p < players.length; p++) {
37
38             // Only Spin if there is not a winner
39             if (winner == "") {
40                 players[p].spin(spinner);
41                 System.out.println(players[p].getName() + " position is: " + players[p].getPosition());
42             } // end if
43
44             // If the player reaches 100, end the While loop
45             if (players[p].getPosition() > 100) {
46                 // Get the winner's name
47                 winner = players[p].getName();
48                 // Set the gameActive to false to exit the While loop
49                 gameActive = false;
50             } // end if
51
52         } // end for
53
54     } // end while
```

We have now completed a Game template where a group of Players spins to get random numbers and advance through 100. The first player to reach 100 wins. In the next Phase we will add the representation of the “Chutes” and “Ladders” to build the complete Chutes and Ladders simulation.

Phase 6: Write the “GameBoard” Class with the Game Logic

We will build on the previous phase and convert the Display class to a complete Chutes and Ladders Game by adding the Chute and Ladder arrays and incorporating logic to have each player check to see if they landed on a chute or a ladder.

1. Declare two Arrays in the Display Class. There are two dimensional arrays. The First dimension represents the number of chutes or ladders. The second dimension holds the first and last value. For Example, Ladder #1's first position is 1. Ladder#1's second position is 38. If a player lands on space 1, they advance to space 38.
2. Modify the Display Class to include the following fields. (We use 'static' because we will use and call these fields within the class. (Insert these at Lines 4 to 8).

```
1
2 public class Display {
3
4     // An Array to hold the Ladder Values
5     public static int ladders[][] = new int[9][2];
6
7     // An Array to hold the Chute Values
8     public static int chutes[][] = new int[10][2];
9
```

3. Now create the Function to assign the Ladder values to the Array. (Note that we are 'hard coding' this function to insert data. Most programs will store this data in an external file or resource.)

```
9
10 // Build the Ladder Array
11 public static void buildLadders() {
12     ladders[0][0] = 1;
13     ladders[0][1] = 38;
14     ladders[1][0] = 4;
15     ladders[1][1] = 14;
16     ladders[2][0] = 9;
17     ladders[2][1] = 31;
18     ladders[3][0] = 21;
19     ladders[3][1] = 42;
20     ladders[4][0] = 28;
21     ladders[4][1] = 84;
22     ladders[5][0] = 36;
23     ladders[5][1] = 44;
24     ladders[6][0] = 51;
25     ladders[6][1] = 67;
26     ladders[7][0] = 71;
27     ladders[7][1] = 91;
28     ladders[8][0] = 80;
29     ladders[8][1] = 100;
30 }
31
```

4. Now write the Function to build the chutes array:

```
32 // Build the Chute Array
33 public static void buildChutes() {
34     chutes[0][0] = 15;
35     chutes[0][1] = 6;
36     chutes[1][0] = 47;
37     chutes[1][1] = 26;
38     chutes[2][0] = 49;
39     chutes[2][1] = 11;
40     chutes[3][0] = 56;
41     chutes[3][1] = 53;
42     chutes[4][0] = 62;
43     chutes[4][1] = 19;
44     chutes[5][0] = 64;
45     chutes[5][1] = 60;
46     chutes[6][0] = 87;
47     chutes[6][1] = 24;
48     chutes[7][0] = 93;
49     chutes[7][1] = 73;
50     chutes[8][0] = 95;
51     chutes[8][1] = 75;
52     chutes[9][0] = 98;
53     chutes[9][1] = 78;
54 }
55
```

- Write the functions to check if a Player landed on a chute or a ladder. These functions also change the players position if they land on a chute or ladder.

```
55
56 // Function to Check if Player landed on a ladder
57 public static void checkLadder(Player p) {
58     for (int i = 0; i < ladders.length; i++) {
59         if (p.getPosition() == ladders[i][0]) {
60             p.setPosition(ladders[i][1]);
61             System.out.println(p.getName() + " Landed on a Ladder!");
62         }
63     }
64 }
65
66 // Function to Check if Player landed on a chute
67 public static void checkChute(Player p) {
68     for (int i = 0; i < chutes.length; i++) {
69         if (p.getPosition() == chutes[i][0]) {
70             p.setPosition(chutes[i][1]);
71             System.out.println(p.getName() + " Landed on a Chute!");
72         }
73     }
74 }
75
```

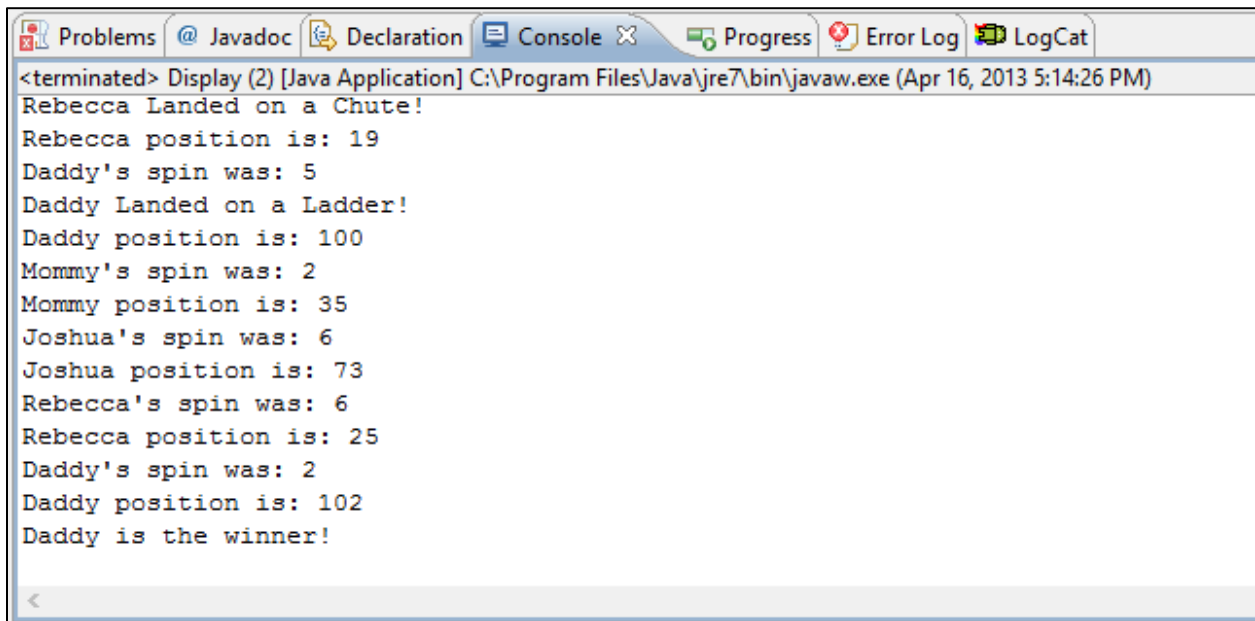
- To build the Chute and Ladders, call the buildChutes() and buildLadders() functions in the main method: (Add lines 82 and 83)

```
76
77 // Test the Player with an Instance
78 // Use the 'main' method to run tests
79 public static void main(String[] args) {
80
81     // Setup the Chutes and Ladders
82     buildLadders();
83     buildChutes();
84
85     // Create 4 Player Instances
86     Player joshua = new Player("Joshua");
87     Player rebecca = new Player("Rebecca");
88     Player daddy = new Player("Daddy");
89     Player mommy = new Player("Mommy");
90
```

7. In the While Loop with the Game Logic, add the `checkLadder()` and `checkChute()` functions. (Lines 118 and 119).

```
109 // While Loop for the Game Logic
110 while (gameActive) {
111
112     // For Loop for Players to take turns
113     for (int p = 0; p < players.length; p++) {
114
115         // Only Spin if there is not a winner
116         if (winner == "") {
117             players[p].spin(spinner);
118             checkLadder(players[p]);
119             checkChute(players[p]);
120             System.out.println(players[p].getName() + " position is: " + players[p].getPosition());
121         } // end if
122
123         // If the player reaches 100, end the While loop
124         if (players[p].getPosition() > 100) {
125             // Get the winner's name
126             winner = players[p].getName();
127             // Set the gameActive to false to exit the While loop
128             gameActive = false;
129         } // end if
130
131     } // end for
132
133 } // end while
134
```

8. That is it! Save and Run the program and your console should show the results of the Chutes and Ladders simulation:



```
<terminated> Display (2) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Apr 16, 2013 5:14:26 PM)
Rebecca Landed on a Chute!
Rebecca position is: 19
Daddy's spin was: 5
Daddy Landed on a Ladder!
Daddy position is: 100
Mommy's spin was: 2
Mommy position is: 35
Joshua's spin was: 6
Joshua position is: 73
Rebecca's spin was: 6
Rebecca position is: 25
Daddy's spin was: 2
Daddy position is: 102
Daddy is the winner!
```

Please note that this is only the model for the Objects and Data for the game – we have not employed an interface for gathering user input or a view model (outside the Print line to console). This lesson is designed to review Java fundamentals, and introduce the architecture we will use when we design Mobile Applications.

Project Scoring Options: (Maximum 100 Points)

Complete the Directions and create a Working Chutes and Ladders Model: 85 Points
(Upload Player, Spinner, and Display Classes to your SkyDrive Folder)
(Upload the copied text from the Console showing the game progress in a text file)
(Upload a Jing of you explaining your code and running the Simulation)

Select another Board Game and Develop:

One Class (Fields, Constructor, Getter, Setter, and Functions) for the Game:
The "Display" Class to test the Class you designed

Up to 8 Points

Select another Board Game and Develop a Complete working Model and Text Display simulating the Game (Including multiple classes representing "Players", "GameBoard", and other game pieces:

Up to 15 points

Refactor and Improve the Java Code and Game Logic for Chutes and Ladders Up to 15 points

(This project was constructed as a "learning project" – so many of the code conventions did not work to maximize efficiency or add "flair" to the Game Display. Show off your Java Skills here and add functionality and your own touch of creativity to the Chutes and Ladders Game)

Project Codes due to your SkyDrive Completed Work Folder by: