```java
import java.util.Scanner;
/**
 * Driver module for Game of Nim.
 *
 * @author CS140 Instructors
 * @version 2/27/2017
 */
public class Project5
{
    public static void main(String args[])
    {
        Scanner console = new Scanner(System.in);
        System.out.print("Enter the minimum number of marbles in your pile: ");
        int min = console.nextInt();

        System.out.print("Enter the maximum number of marbles in your pile: ");
        int max = console.nextInt();
        GameOfNim game = new GameOfNim(min, max);

        game.play();
    }
}
```

98/100

```
import java.util.Scanner;
import java.util.Random;

/**
 * A simple game of Nim.
 *
 * @author Phil Fevry
 * @version 1.0
 */
public class GameOfNim
{
    private int min, max, marbleCount;
    private boolean firstMove, smartMode;
    private Player currentPlayer;

    public GameOfNim(int min, int max)
    {
        this.min = min;
        this.max = max;
        firstMove = true;
        marbleCount = 0;
    }

    public void play()
    {
        // get stuff ready
        marbleCount = newMarbleCount();

        // start the fun
        output("\nGame begins!\n");
        output("Initially there are [" + marbleCount + "] marbles in the pile.\n
");

        do {
            if (winner()) {
                gameOver(currentPlayer);
                break;
            }
            takeTurns();
        } while (marbleCount > 0);

        if (marbleCount == 0)
            gameOver(getTurn()); // currentPlayer lost
    }

    private void takeTurns()
    {
        if (!winner())
        {
            currentPlayer = getTurn();
            output("\n" + currentPlayer + "'s Turn\n");
            applyMove(currentPlayer);
        } else {
            gameOver(currentPlayer); // currentPlayer wins
        }
    }

    private Player getTurn()
    {
        if (firstMove)
        {
            firstMove = false;
```

-2

Should tell you whether computer is playing in smart mode

```
            Random randomizer = new Random();
            // Randomize startMode and who goes first
            smartMode = randomizer.nextInt(1) == 0 ? true:false;
            return (randomizer.nextInt(1) == 0) ? Player.COMPUTER : Player.HUMAN
;
        } else {
            return (currentPlayer == Player.HUMAN) ? Player.COMPUTER : Player.HU
MAN;
        }
    }

    private void applyMove(Player p)
    {
        int marblesRemoved = 0;

        if (p == Player.HUMAN)
        {
            output("Enter number of marbles to remove: ");
            Scanner console = new Scanner(System.in);
            int n = console.nextInt();
            while (!legalMove(n))
            {
                output("Enter number of marbles to remove: ");
                n = console.nextInt();
            }

            marblesRemoved = n;
        } else {
            marblesRemoved = computerDecision();
        }

        marbleCount -= marblesRemoved;
        output("\t" + currentPlayer +" removed " + marblesRemoved + " marble(s).
\n");

        if (currentPlayer == Player.COMPUTER)
        {
            if (marbleCount != 1)
                output("\n[" + marbleCount + "] marbles remain in the pile.\n");
            else
                output("\n[" + marbleCount + "] marble remains in the pile.\n");
        }
    }

    private void gameOver(Player winner)
    {
        output("\n" + winner + " wins!\n");

        // play again?
        output("\nDo you want to play the game again? Enter Yes or No: ");
        Scanner console = new Scanner(System.in);
        if (console.next().toUpperCase().equals("YES"))
        {
            output("\nEnter the minimum number of marbles in your pile: ");
            min = console.nextInt();

            output("Enter the maximum number of marbles in your pile: ");
            max = console.nextInt();

            firstMove = true;

            play();
        } else {
```

```
            output("Thanks for playing!");
        }
    }

    private int newMarbleCount()
    {
        Random randomizer = new Random();
        return randomizer.nextInt(max + 1 − min) + min;
    }

    private boolean legalMove(int n)
    {
        if (n > marbleCount)
        {
            output("There are less than " + n + " marbles on the board!\n");
            return false;
        }

        if (n > (marbleCount/2))
        {
            output("You can’t pick up more than half of total marbles at once!\n
");
            return false;
        }

        return true;
    }

    private int computerDecision()
    {
        Random randomizer = new Random();
        int decision = randomizer.nextInt((marbleCount/2)) + 1; // initialized w
ith any legal move

        // change decision based on certain conditions
        if (smartMode)
        {
            // the best position to be in is the one where the marbleCount is
            // one less than a number which is a power of two after your turn.
            // in smartMode the computer attempts to always be in this position

            // is marbleCount a number which is a power of two?
            boolean marbleCountIsPowOfTwo = isPowOfTwo(marbleCount);
            //is marbleCount one less than a number which is a power of two?
            boolean marbleCountIsOneLessPowOfTwo = isPowOfTwo(marbleCount+1) ? t
rue : false;

            if (!marbleCountIsPowOfTwo)
            {
                if (!marbleCountIsOneLessPowOfTwo)
                {
                    // find the next number that is one less than a power of two
                    for (int i = marbleCount; i > 0; i−−)
                    {
                        if (isPowOfTwo(i+1))
                        {
                            decision = (marbleCount − i);
                            break;
                        }
                    }
                }
            } else {
                // marbleCount is a power of two so just remove 1
```

```
                decision = 1;
            }
        }
        return decision;
    }

    private boolean isPowOfTwo(int n)
    {
        // uses ’bitwise’ operation ’&’ which finds the power of two nonzero num
bers
        return (n > 0 && (marbleCount & marbleCount −1) == 0) ? true : false;
    }

    private boolean winner()
    {
        return (marbleCount == 1 || marbleCount == 0);
    }

    private void printMarbleCount()
    {
        output("\tCurrent number of marbles: " + marbleCount + "\n");
    }

    private static void output(String s)
    {
        System.out.print(s);
    }
}
```

Discussion Log
Assignment: Project 5
Name: Phil Fevry

Time taken: 3 days @ around 5 hours each (20 hours)

What I learned:
− I gained more understanding designing and organizing class methods.
− I gained valuable practice on creating loops making me more proficient.
− I solidified my understanding of the phrase "dont reinvent the wheel" and found an existing algorithm for finding a number which is a power of 2 online (1)

Difficulties I faced:
− Figuring out the math and logic for smartMode is the only difficulty I faced.

Resources Used:
(1) http://www.programcreek.com/2014/07/leetcode-power-of-two-java/
(2) https://en.wikipedia.org/wiki/Nim

```
commit dae6b6ae606994051abc1d195d5e098294ce098c
Author: Phil Fevry <pfevry@worcester.edu>
Date:   Tue Apr 4 20:44:38 2017 −0400

    Project 5 Final Version

commit e74d1c9b6501bb55fbd8c41bb104523bec036d3c
Author: Aparna Mahadev <amahadev@worcester.edu>
Date:   Tue Mar 14 13:41:27 2017 −0400

    Project 5
```