```java
import java.io.BufferedReader;
import java.io.*;
import java.io.InputStreamReader;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.*;
import java.util.regex.Pattern;

public class Game1024Alex {


    private static Scanner input = new Scanner (System.in);
    private static BufferedReader input2 = new BufferedReader(new InputStreamReader(System.in));
    private static int verticalCells; // determine number of vertical cells
    private static int horizontalCells; // determine number of horizontal cells
    private static int cellValue = 1; // used for setting vallue of cells
    private static int cellBoardCopy [][]; // use to compare array with Arrays.deepEquals board values
    private static int cellBoard [][]=new int [horizontalCells][verticalCells]; // board values
    private static Random random1=new Random(); // generates random numbers
    private static int initialTile = 0;
    private static int score=0;
    private static boolean outcome = false;
    private static String textBufferedReader;
    private static int intBufferedReader=0;
    private static boolean catMode = false;
    private static int catLives = 9;
    private static ArrayList <Integer> replacementCells = new ArrayList <Integer>(); // holds 11 cells from 1-2048 for the cat Mode
    private static ArrayList <Integer> tempReadCells = new ArrayList <Integer>(); // used to recreate the board from a saved file
    private static boolean catModeFirstRun = true;
    private static ArrayList <String> readList = new ArrayList <String>(); // stores data in save file
    private static File savedGame = new File ("save.csv");
    private static boolean gameOver=false;
    private static boolean run=true;
    private static boolean winMessage = true;
    private static boolean firstLoad = true;
    private static boolean skipBlock = false;



    public static void main(final String[] args) throws Exception {
        // TODO Auto-generated method stub
        if (savedGame.exists() == true && firstLoad == true) {
            while(intBufferedReader <1 || intBufferedReader>2) {
            try {
                System.out.println("Would you like to continue your old game or start a new one ?");
                System.out.println("\n1. New Game");
                System.out.println("2. Continue");
                textBufferedReader = input2.readLine();
                intBufferedReader = Integer.parseInt(textBufferedReader);
            } catch (Exception e) {
                System.out.println("Please enter digit 1 or 2");
            }
            if(intBufferedReader == 2) {
```

```
            firstLoad = false;
        loadGame();
        }else if(intBufferedReader == 1){
            try {
                firstLoad = false;
                System.out.println("" + "\n*****************************" + "\n*    WELCOME TO 1024/2048    *"
                        + "\n*Choose play area 4-4 to 8-8 *" + "\n*    Enter 4,5,6,7 or 8      *"
                        + "\n*****************************");

                while (true) {
                    try {
                        textBufferedReader = input2.readLine();
                        verticalCells = Integer.parseInt(textBufferedReader);
                    } catch (Exception e) {
                        System.out.println("" + "Error please enter a digit" + "\n--------------------------");
                    }
                    horizontalCells = verticalCells;
                    cellBoard = new int[horizontalCells][verticalCells];
                    if (verticalCells < 4 || verticalCells > 8) {
                        System.out.println("Choose number between 4-8");
                    } else
                        break;
                }
            } catch (Exception e) {
                System.out.println("Error deleting File Save may Corupt");
            }
        }
    } else {
        try {
            firstLoad = false;
            System.out.println("" + "\n*****************************" + "\n*    WELCOME TO 1024/2048    *"
                    + "\n*Choose play area 4-4 to 8-8 *" + "\n*    Enter 4,5,6,7 or 8      *"
                    + "\n*****************************");

            while (true) {
                try {
                    textBufferedReader = input2.readLine();
                    verticalCells = Integer.parseInt(textBufferedReader);
                } catch (Exception e) {
                    System.out.println("" + "Error please enter a digit" + "\n--------------------------");
                }
                horizontalCells = verticalCells;
                cellBoard = new int[horizontalCells][verticalCells];
                if (verticalCells < 4 || verticalCells > 8) {
                    System.out.println("Choose number between 4-8");
                } else
                    break;
            }
        } catch (Exception e) {
            System.out.println("Error deleting File Save may Corupt");
        }
    }
```

```
            while (outcome == false) {
                gameControl();
                if (gameOver == false) {
                    saveGame();
                }
            }



    }

    private static void saveGame () throws Exception {
        System.out.println("gameSave");
        FileWriter fileW= new FileWriter(savedGame, false); // the true keyword will make the writer add next line without deleting th
e old one if set to false it will replace lines
        PrintWriter printW = new PrintWriter(fileW);
        printW.print("GridSize"+verticalCells+",\n");
        for (int i =0; i < cellBoard.length; i++) {
            for (int j=0; j < cellBoard[i].length; j++) {
                printW.print("Cell"+cellBoard[i][j]+",");
            }
            printW.print("\n");
        }

        printW.print("Score"+score+",");
        printW.print("CatLives"+catLives+",\n");
        for (int i=0; i < replacementCells.size(); i++ ) {
            printW.print("ReplaceCell"+replacementCells.get(i)+",");
        }
        printW.print("\n");
        if (catMode==true) {
            printW.print("catMode=true,");
        }else {
            printW.print("catMode=false,");
        }
        if (catModeFirstRun==true) {
            printW.print("catModeFirstRun=true,");
        }else {
            printW.print("catModeFirstRun=false,");
        }
        if (run==true) {
            printW.print("run=true,");
        }else {
            printW.print("run=false,");
        }
        if (winMessage==true) {
            printW.print("winMessage=true,");
        }else {
            printW.print("winMessage=false,");
        }
        if (skipBlock==true) {
            printW.print("skipBlock=true,");
        }else {
            printW.print("skipBlock=false,");
```

```
    }
    printW.print("\ninitialTile"+initialTile+",");
    printW.close();

}



private static void loadGame() throws Exception {
    System.out.println("gameLoad");
    input2 = new BufferedReader(new FileReader("save.csv"));

    String line = "";

    while ((line = input2.readLine()) != null) { // read file
        String[] str = line.split(",");
        for (int i = 0; i < str.length; i++) {
            readList.add(str[i]);
        }
    }

    for (int j = 0; j < readList.size(); j++) {

        line = readList.get(j);
        Object[] splitLine = splitData(line);
        String text = Objects.toString(splitLine[0]);
        String textNumber = Objects.toString(splitLine[1]);
        int number = Integer.parseInt(textNumber);

        switch (text) {
        case "GridSize":
            verticalCells = number;
            horizontalCells = verticalCells;
            cellBoard = new int[horizontalCells][verticalCells];
            for (int k = 0; k < readList.size(); k++) {
                line = readList.get(k);
                Object[] splitLine2 = splitData(line);
                text = Objects.toString(splitLine2[0]);
                textNumber = Objects.toString(splitLine2[1]);
                number = Integer.parseInt(textNumber);
                if (text.equals("Cell")) {
                    tempReadCells.add(number);
                }
            }
            int getNext = 0;
            for (int a = 0; a < cellBoard.length; a++) {
                for (int b = 0; b < cellBoard[a].length; b++) {
                    cellBoard[a][b] = tempReadCells.get(getNext);
                    getNext++;
                }
            }
            break;
        case "Score":
            score = number;
```

```
            break;
        case "CatLives":
            catLives = number;
            break;
        case "ReplaceCell":
            replacementCells.add(number);
            break;
        case "initialTile":
            initialTile = number;
            break;
        }

        if(Pattern.matches("catModeFirstRun=true", line)) {
            catModeFirstRun =true;
        }else {
            catModeFirstRun =false;
        }
        if(Pattern.matches("catMode=true", line)) {
            catMode =true;
        }else {
            catMode =false;
        }
        if(Pattern.matches("run=true", line)) {
            run =true;
        }else {
            run =false;
        }
            if(Pattern.matches("skipBlock=true", line)) {
                skipBlock = true;
            }else {
                skipBlock = false;
            }


            input2 = new BufferedReader(new InputStreamReader(System.in));

        }

        for (int i=0; i < replacementCells.size(); i++) {
            System.out.println(replacementCells.get(i));
        }
        skipBlock = true;
        gameControl();
    }

    public static Object[] splitData(String line) { // method an array of obkects to split a string and return a string and an integer

        String start = "";
        String end = "";
        int number = 0;

        for (int i = 0; i < line.length(); i++) {
            if (Character.isDigit(line.charAt(i))) {
                start = line.substring(0, i);
```

```
                end = line.substring(i, line.length());
                number = Integer.parseInt(end);
                break;
            }
        }
        return new Object[] { start, number };
    }



    private static void catMode ()throws Exception {
        catMode = true;
        skipBlock = true;
        int intBufferedReader2=0;

        do {
        System.out.println(""
                + "\n***********************************"
                + "\n*                                 *"
                + "\n*       Cat Mode(9) Entered       *"
                + "\n*                                 *"
                + "\n***********************************");
        gridBuilder();
//      do {
        System.out.println(textBufferedReader);
        System.out.println(intBufferedReader);
            try {
        System.out.println("-- Enter cell number from 1 to "+((cellBoard.length)*(cellBoard[0].length))+" and amend values --");
        textBufferedReader = input2.readLine();
        intBufferedReader = Integer.parseInt(textBufferedReader);
            }catch (Exception e){
                System.out.println("\nPlease enter a valid number!!!\n");
            }


//      }while (intBufferedReader < 1 || intBufferedReader >((cellBoard.length)*(cellBoard[0].length)));

        int cellNumber = intBufferedReader;
        int count=1;
        for (int i=0; i < cellBoard.length; i++) {
            for (int j=0; j < cellBoard[i].length; j++) {
                if (cellNumber == count) {
                    System.out.println(cellNumber+" "+ count);
                    for (int k=0; k < replacementCells.size(); k++) {
                        System.out.println((k+1)+" -- " + replacementCells.get(k));
                    }
                    do {
                        try {
                    System.out.println("Choose number to replace the contents of cell "+cellNumber+" from 1 to "+replacementCells.size
());
                    textBufferedReader = input2.readLine();
                    intBufferedReader = Integer.parseInt(textBufferedReader);
                        }catch (Exception e){
                            System.out.println("\nPlease enter a valid number!!!\n");
```

```
                    }
                }while (intBufferedReader < 0 && intBufferedReader > replacementCells.size());
                cellBoard[i][j] = replacementCells.get(intBufferedReader-1);
                replacementCells.remove(intBufferedReader-1);
                catLives--;
            }
            count++;
        }
    }

    gridBuilder();
    System.out.println("         &                    &                   \r\n" +
            "        &&&&&             &&&&&                   \r\n" +
            "         &&&&&&&&&&&&&&&&&&&&&                    \r\n" +
            "          &&&&&&&&&&&&&&&&&&&             &&&&&   \r\n" +
            "         &&&&&&&&&&&&&&&&&&&&        &&&&&&&&&&&&&\r\n" +
            "         &&&&& O (&&, O &&&&&      &&&&        &&&\r\n" +
            "          &&&&&&&&&&&&&&&&&&&     &&&&    &&&&&&&&\r\n" +
            "           &&&&&&____&&&&&&&       &&&             \r\n" +
            "             &&&&&&&&&&&&&        &&&             \r\n" +
            "              &&&&&&&&&&&&        &&&&            \r\n" +
            "             &&&&&&&&&&&&&&         &&&&          \r\n" +
            "            &&&&&&&&&&&&&&&&&&       &&&&&        \r\n" +
            "            &&&&&&&&&&&&&&&&&&&&&      &&&&&&     \r\n" +
            "            &&&&&&&&&&&&&&&&&&&&&&&       &&&&&   \r\n" +
            "           &&&&&&&&&&&&&&&&&&&&&&&&&&       &&&&  \r\n" +
            "            &&&&&&&&&&&&&&&&&&&&&&&&&&       &&&& \r\n" +
            "            &&&&&&&&&&&&&&&&&&&&&&&&&&&       &&& \r\n" +
            "             &&&&&&&&&&&&&&&&&&&&&&&&&&&      &&& \r\n" +
            "              &&&&&&&&&&&&&&&&&&&&&&&&&&     &&&& \r\n" +
            "               &&&&&&&&&&&&&&&&&&&&&&&&&     &&&  \r\n" +
            "                &&&&&&&&&&&&&&&&&&&&&&&&    &&&&  \r\n" +
            "                &&&&&&&&&&&&&&&&&&&&&&&&    &&&   \r\n" +
            "                &&&&&&&&&&&&&&&&&&&&&&&&  &&&&    \r\n" +
            "              &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&     \r\n" +
            "              &&&&&&&&&&&&&&&&&&&&&&&&&&&&&      ");


    System.out.println("\nYou have "+catLives+" lives left"+" meow :)");
    System.out.println("Amend another cell Y/N ?");
    textBufferedReader = input2.readLine();
    }while (textBufferedReader.equalsIgnoreCase("Y"));
    catMode = false;
}

private static void gameCondition() throws Exception {
    System.out.println("gameCondition");
    outcome = true;
    outerloop:
    for (int i=0; i < cellBoard.length; i++) {
        for (int j=0; j < cellBoard[i].length; j++) {
            if (cellBoard[i][j] == 0) {
                outcome = false;
            }else if(cellBoard[i][j] >= 1024 && winMessage == true) {
```

```
            System.out.println(""
                    + "\n**********************************"
                    + "\n*  Congratulations YOU WON !!!   *"
                    + "\n*        Keep Playing !!!        *"
                    + "\n**********************************\n");
            winMessage = false;

        }
    }
}

if (outcome == true) {
    System.out.println(""
            + "\n****************************************"
            + "\n*  GAME OVER NO EMPTY SPACE LEFT !!!   *"
            + "\n****************************************\n");
    while (savedGame.exists() == true) {
    try {
    Files.delete(Paths.get("save.csv"));
    gameOver = true;
    }catch (Exception e){
            System.out.println("File Coruption");
        }
    }
}
```

```
}

private static void gameControl () throws Exception{
    System.out.println("gameControl");
    String move;
    gridBuilder();
    System.out.println("Score: "+score);
    gameCondition();
    if(outcome == false) {
    boolean unrecognizedComand = false;
    do {

        if (unrecognizedComand == true)
            System.out.println("\n (Unknown input) \n");


    System.out.println(""
            + "\n-------------------------"
            + "\nPres W to move up"
            + "\nPres S to move down"
            + "\nPres A to move left"
            + "\nPres D to move right"
            + "\n-------------------------"
            + "\nType \"Cat mode\" to test!"
            + "\nType \"Exit\" to exit game!"
```

```
            + "\n-------------------------");

    move = input.nextLine();
    unrecognizedComand = true;
        } while (!move.equalsIgnoreCase("w") && !move.equalsIgnoreCase("a") && !move.equalsIgnoreCase("s")
                && !move.equalsIgnoreCase("d") && !move.equalsIgnoreCase("Cat Mode") && !move.equalsIgnoreCase("Exit"));
    if (move.equalsIgnoreCase("Cat Mode")) {
        catMode();
    }
    else if (move.equalsIgnoreCase("Exit")) {
        outcome = true;
    }else {
    gameMovement(move);
    }
    }

}

private static void gameMovement (String input) {
    System.out.println("gameMovement");
    String direction = input;

    int tempA, tempB;

    gamemove: while (true) {

        cellBoardCopy = cellBoard.clone(); // clone array for comratison of changes

if (direction.equalsIgnoreCase("s")) {

    for (int i = 0; i < (cellBoard.length - 1); i++) {
        for (int j = 0; j < cellBoard[i].length; j++) {
            if (cellBoard[i][j] > 0) {
                tempA = cellBoard[i][j];
                tempB = cellBoard[i + 1][j];
                if (cellBoard[i + 1][j] == 0) { // move thorough empry space
                    cellBoard[i][j] = tempB;
                    cellBoard[i + 1][j] = tempA;
                    i=0;
                    j=0;
                } else if (cellBoard[i][j] == cellBoard[i + 1][j]) { // merge same numbers
                    cellBoard[i][j] = 0;
                    cellBoard[i + 1][j] *= 2;
                    score += cellBoard[i + 1][j];
                    i=0;
                    j=0;
                }
            }
        }
    }
} else if (direction.equalsIgnoreCase("W")) {
    for (int i = cellBoard.length - 1; i > 0; i--) {
        for (int j = 0; j < cellBoard[i - 1].length; j++) {
            if (cellBoard[i][j] > 0) {
```

```
                tempA = cellBoard[i][j];
                tempB = cellBoard[i - 1][j];
                if (cellBoard[i - 1][j] == 0) {
                    cellBoard[i][j] = tempB;
                    cellBoard[i - 1][j] = tempA;
                    i=cellBoard.length - 1;
                    j=0;
                } else if (cellBoard[i][j] == cellBoard[i - 1][j]) {
                    cellBoard[i][j] = 0;
                    cellBoard[i - 1][j] *= 2;
                    score += cellBoard[i - 1][j];
                    i=cellBoard.length - 1;
                    j=0;
                }
            }
        }

    }
} else if (direction.equalsIgnoreCase("d")) { // fix movements with i j reset
    for (int i = 0; i < cellBoard.length; i++) {
        for (int j = 0; j < cellBoard[i].length - 1; j++) {
            if (cellBoard[i][j] > 0) {
                tempA = cellBoard[i][j];
                tempB = cellBoard[i][j + 1];
                if (cellBoard[i][j + 1] == 0) {
                    cellBoard[i][j] = tempB;
                    cellBoard[i][j + 1] = tempA;
                    i=0;
                    j=0;
                } else if (cellBoard[i][j] == cellBoard[i][j + 1]) {
                    cellBoard[i][j] = 0;
                    cellBoard[i][j + 1] *= 2;
                    score += cellBoard[i][j + 1];
                    i=0;
                    j=0;
                }
            }
        }
    }
} else if (direction.equalsIgnoreCase("a")) {
    for (int i = 0; i < cellBoard.length; i++) {
        for (int j = cellBoard[i].length - 1; j > 0; j--) {
            if (cellBoard[i][j] > 0) {
                tempA = cellBoard[i][j];
                tempB = cellBoard[i][j - 1];
                if (cellBoard[i][j - 1] == 0) {
                    cellBoard[i][j] = tempB;
                    cellBoard[i][j - 1] = tempA;
                    i=0;
                    j = cellBoard[i].length - 1;
                } else if (cellBoard[i][j] == cellBoard[i][j - 1]) {
                    cellBoard[i][j] = 0;
                    cellBoard[i][j - 1] *= 2;
                    score += cellBoard[i][j - 1];
```

```
                            i = 0;
                            j = cellBoard[i].length - 1;
                        }
                    }
                }
            }
        }
        if (Arrays.deepEquals(cellBoard, cellBoardCopy) == true) {
            break;

        }

    }

}

private static void gameEngine() { // populates board values

    if (catModeFirstRun == true) {
        for (int i=1; i <= 2048; i*=2 ) {
            replacementCells.add(i);
        }
        catModeFirstRun = false;
    }
    if (skipBlock == false) {
    System.out.println("gameEngine");
    int randomTile;
outerloop: // label to brake out from nested loops
while (true) {
    for (int i = 0; i < cellBoard.length; i++) {
        for (int j = 0; j < cellBoard[i].length; j++) {
            if (cellBoard[i][j] == 0) {
                run = false;
                randomTile = random1.nextInt(100);
                if (randomTile > 97) {
                    cellBoard[i][j] = 1;
                    initialTile++;
                    if (initialTile >= 2) { // set initial 2 tiles and 1 after
                        break outerloop;
                    }
                }
            }

        }

    }
    if (run == true) {
        break;
    }

}
}
skipBlock=false;
```

```
}

private static void gridBuilder() { // builds grid visually
    System.out.println("gridBuilder");
    if (catMode == false) {
    gameEngine(); // TESTING
    }
    int row = 0;
    int col = 0;
    int cellCount=1;


    for (int i = 0; i < verticalCells * 4; i++) {
        if (i % 4 == 0) {
            for (int j = 0; j <= horizontalCells * 9; j++) {
                if (j % 9 == 0) {
                    System.out.print("+");
                } else
                    System.out.print("-");
            }
        } else
            for (int j = 0; j <= horizontalCells * 9; j++) {
                if (j % 9 == 0) {
                    System.out.print("|");
                } else if (i % 4 == 2 && j % 3 == 0) {

                    cellValue = cellBoard[col][row];
    if (cellValue == 0) {
        System.out.print("      ");
    } else if (cellValue < 10) {
        System.out.print("  " + cellValue + "   ");
    } else if (cellValue < 100) {
        System.out.print(" " + cellValue + "   ");
    } else if (cellValue < 1000) {
        System.out.print(" " + cellValue + "  ");
    } else if (cellValue < 10000) {
        System.out.print(cellValue + "  ");
    } else
        System.out.print("ERROR ");
    j += 5;
    row++;
    if (row == horizontalCells) {
        row = 0;
        col++;
    }
} else if ((i % 4 == 1 && j % 9 == 1) && catMode == true) {
    if (cellCount < 10) {
    System.out.print(cellCount);
    cellCount++;
    }else if (cellCount >= 10) {
    System.out.print(cellCount);
    cellCount++;
    j++;
    }
```

```
                    }else {
                        System.out.print(" ");
                    }
                }
            System.out.println();
            if ((i + 1) == verticalCells * 4) {
                for (int j = 0; j <= horizontalCells * 9; j++) {
                    if (j % 9 == 0) {
                        System.out.print("+");
                    } else
                        System.out.print("-");
                }
            }
        }
        System.out.println();
    }


}
```