```
import java.util.Arrays;
import java.util.Scanner;
import java.util.InputMismatchException;

/**
 * Yahtzee Game
 *
 * @1.0.0
 */
public class Yahtzee
{
    Scanner scanner = new Scanner(System.in);
    Die6 die1 = new Die6();
    Die6 die2 = new Die6();
    Die6 die3 = new Die6();
    Die6 die4 = new Die6();
    Die6 die5 = new Die6();
    private int[] scoreUpper = new int[6];
    private int upperTotal;
    private int[] scoreOfAKind = new int[3];
    private int fullHouse;
    private int NUM_DIE = 6;
    private int[] straight = new int[2];
    private int lowerTotal;
    private int grandTotal;
    private int Chance = 0;
    private String input;
    private static final int NUM_DICE = 5;
    private static final int NUM_ROUNDS = 13;
    private int round;
    private int score;

    /**
     * Constructor for the Yahtzee class. Initializes the game's
     * round number to 1, and rolls all dice.
     */
    public Yahtzee() {
        round = 1;
        rollAll();
    }

    /**
     * The main method for the Yahtzee class.
     */
    public static void main(String[] args) {
        Yahtzee yahtzee = new Yahtzee();
        yahtzee.play();
    }

    /**
     * A class representing a game of Yahtzee.
     * Allows users to roll all dice, reroll some of them, and
     * then select a category to score their dice into. Scores are
     * kept track of for both upper section categories and lower section
categories.
     */
    public void play() {
        int turn = 0;
        int scoreRound = 0;
        boolean rolled = false;
```

```
        boolean scoreMarked = false;
        while(round <= NUM_ROUNDS) {
            System.out.print("\n[ "+toString()+" ]");
            System.out.println("\n{ - 0 - Quit | 4 - Next Turn }");
            System.out.println("{ -- roll - Rolls Specified Dice |
rollAll - Rolls All Dice }");
            System.out.println("{ --- 1 - Mark Uppersection | 2 - Mark
Lowersection | 3 - Getscores }\n");
            input = scanner.nextLine();

            switch(input) {
                case "rollAll":
                    if(!rolled) {
                        rollAll();
                        turn++;
                        System.out.println("Turn:"+turn);
                    } else {
                        System.out.println("(-Out of Turns)");
                    }
                    break;

                case "roll":
                    if(!rolled) {
                        System.out.println("{0 - back to the main
menu}");
                        System.out.println("Enter The Die You Wish To
Roll Separated By a Space");
                        String cutToPieces = scanner.nextLine();

                        if(cutToPieces.equals("0")){
                            System.out.println("Going back to the main
menu");
                            break;
                        }

                        String[] pieces = cutToPieces.split(" ");
                        int[] dicearray = new int[pieces.length];
                        for(int i = 0; i < dicearray.length; i++) {
                            dicearray[i] = Integer.parseInt(pieces[i]);
                        }
                        roll(dicearray);
                        turn++;
                        System.out.println("Turn:"+turn);
                    } else {
                        System.out.println("(-Out of Turns)");
                    }
                    break;

                    //Done
                case "1":
                    if (scoreMarked) {
                        System.out.println("(-Score already marked in
this round.)");
                    } else {
                        System.out.println("{0 - back to the main
menu}");
                        System.out.println("Select a category (1-6)");

                        int categoryUpper = scanner.nextInt();
```

```
                        if(categoryUpper == 0) {
                            System.out.println("Going back to the main
menu");
                            break;
                        }

                        scanner.nextLine();
                        System.out.println(scoreUpper(categoryUpper));
                        scoreMarked = true;
                        scoreRound += scoreUpper(categoryUpper);
                    }
                    break;

                case "2":
                    if (scoreMarked) {
                        System.out.println("(-Score already marked in
this round.)");
                    } else {
                        System.out.println("{0 - back to the main
menu}");
                        System.out.println("1 - Score Of A Kind");
                        System.out.println("2 - Score FullHouse");
                        System.out.println("3 - Score Straights");
                        System.out.println("4 - Score Chance");
                        int cateogoryLower = scanner.nextInt();
                        scanner.nextLine();
                        switch (cateogoryLower) {
                            case 0:
                                System.out.println("Going back to the
main menu");
                                break;
                            case 1:
                                System.out.println("{0 - back to the main
menu}");
                                System.out.println("Please enter 3-5");

                                int markScoreKind = scanner.nextInt();
                                scanner.nextLine();

                                switch(markScoreKind) {
                                    case 0:
                                        System.out.println("Going back to
the main menu");
                                        break;
                                    case 3:
                                        System.out.println("3 of a Kind:
"+scoreOfAKind(3));
                                        scoreRound += scoreOfAKind(3);
                                        break;
                                    case 4:
                                        System.out.println("4 of a kind:
"+scoreOfAKind(4));
                                        scoreRound += scoreOfAKind(4);
                                        break;
                                    case 5:
                                        System.out.println("Yahtzee:
"+scoreOfAKind(5));
                                        scoreRound += scoreOfAKind(5);
                                        break;
                                }
```

```
                                break;
                            case 2:
                                System.out.println(fullHouse());
                                scoreRound += fullHouse();
                                break;
                            case 3:
                                System.out.println("{0 - back to the main
menu}");
                                System.out.print("Please enter 4 or 5:
");

                                int markScoreStraight =
scanner.nextInt();
                                scanner.nextLine();

                                switch(markScoreStraight) {
                                    case 0:
                                        System.out.println("Going back to
the main menu");
                                        break;
                                    case 4:
                                        System.out.println("small
straight: "+straight(4));
                                        scoreRound += straight(4);
                                        break;
                                    case 5:
                                        System.out.println("large
straight: "+straight(5));
                                        scoreRound += straight(5);
                                        break;
                                }
                                break;
                            case 4:
                                System.out.println("Chance :"+Chance());
                                scoreRound += Chance();
                                break;
                        }

                        scoreMarked = true;
                    }
                    break;
                    //Done
                case "3":
                    System.out.println("{0 - back to the main menu}");
                    System.out.println("1 - Upper Scores");
                    System.out.println("2 - Lower Scores");
                    System.out.println("3 - Totals");
                    int type = scanner.nextInt();
                    scanner.nextLine();
                    switch(type) {
                        case 0:
                            System.out.println("Going back to the main
menu");
                            break;
                        case 1:
                            System.out.println(getScoreUpper(1));
                            System.out.println(getScoreUpper(2));
                            System.out.println(getScoreUpper(3));
                            System.out.println(getScoreUpper(4));
                            System.out.println(getScoreUpper(5));
```

```
                        System.out.println(getScoreUpper(6));
                        break;
                    case 2:
                        System.out.println(getScoreOfAKind(3));
                        System.out.println(getScoreOfAKind(4));
                        System.out.println(getScoreOfAKind(5));
                        System.out.println(getFullHouse());
                        System.out.println(getStraight(4));
                        System.out.println(getStraight(5));
                        System.out.println(getChance());
                        break;
                    case 3:
                        System.out.println(getUpperTotal());
                        System.out.println(getLowerTotal());
                        System.out.println(getGrandTotal());
                        break;
                }
                break;
                // Done
            case "4":
                if(!rolled) {
                    System.out.println("You are required to roll");
                } else {
                    round++;
                    System.out.println("Round: "+round);
                    scoreMarked = false;
                    rolled = false;
                    turn  = 0;
                    scoreRound = 0;
                }
                break;
        }

        if(scoreRound == 0) {
            scoreMarked = false;
        }

        if(turn >= 3) {
            rolled = true;
        }

        if(input.equals("0")){
            break;
        }
    }
}

/**
 * Method rolls all dice.
 * Uses an object array to store the dice objects,
 * then iterates over the array and calls the roll method
 * on each object to roll each die.
 */
public void rollAll() {
    Die6[] dice = new Die6[]{die1, die2, die3, die4, die5};
    //iterates over object array and calls roll method
    for (Die6 die : dice) {
        die.roll();
    }
}
```

```
    /**
     * Method rolls the dice specified by an integer array.
     * 1-indexed, not 0-indexed.
     * For example, to roll the first, third, and fifth dice, pass an
integer array {1, 3, 5}
     * @param diceToRoll an array of integers representing the indices of
the dice to be rolled
     */
    public void roll(int[] diceToRoll) {
        for (int i : diceToRoll) {
            switch (i) {
                case 1:
                    die1.roll();
                    break;
                case 2:
                    die2.roll();
                    break;
                case 3:
                    die3.roll();
                    break;
                case 4:
                    die4.roll();
                    break;
                case 5:
                    die5.roll();
                    break;
            }
        }
    }

    /**
     * Returns a string of the five dice values.
     *
     * @return a string of the five dice values.
     */
    public String toString() {
        return("Dice Values: " + die1.value + " " + die2.value + " " +
die3.value + " " + die4.value + " " + die5.value);
    }

    /**
     * Calculates the score for the upper section of the Yahtzee game.
     * Only accepts scores between 1-6.
     * Returns the stored value.
     * Stores the calculated score in the scoreNum and then stores it in
the index of the scoreUpper array.
     * @param score an integer representing the score value to be
calculated for the upper section
     * @return an integer score for the upper section based on the
parameter
     */
    public int scoreUpper(int score) {
        int scoreNum = 0;
        int[] dice = new
int[]{die1.value,die2.value,die3.value,die4.value,die5.value};
        try{
            if(score > 6 || score < 1) {
                throw new ArrayIndexOutOfBoundsException("Please enter 1-
6");
```

```
            }
            //if there's a stored value in the array stops the
calculations
            if (scoreUpper[score-1] != 0) {
                return scoreUpper[score-1];
            }
            //iterates over dice, calculates Uppersection and stores
values in int scoreNum, and then stores it in the instances array
scoreUpper[]
            for(int i: dice) {
                if(i == score) {
                    scoreNum += score;
                    scoreUpper[score-1] = scoreNum;
                }
            }
            //catches ArrayIndexOutOfBoundsException for 1 > score > 6
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Error: " + e.getMessage());
        }
        return scoreNum;
    }

    /**
     * Method calculates and returns the score of a specified type.
     * @param type an integer representing the type of score to be
calculated, where 3 represents to "3 of a kind",
     * 4 represents to "4 of a kind", and 5 represenys to "Yahtzee".
     * @return an integer representing the score of the given type.
     * @throws ArrayIndexOutOfBoundsException if the type is not within
the valid range of 3-5.
     */
    public int scoreOfAKind(int type) {
        int[] counts = new int[6];
        int score = 0;
        boolean yahtzeeBonus = false;
        int[] dice = new
int[]{die1.value,die2.value,die3.value,die4.value,die5.value};
        try{
            if(type > 5 || type < 3) {
                throw new ArrayIndexOutOfBoundsException("Please enter 3-
5");
            }
            //checks for already initalized scoreOfAKind[] indexes
            if (scoreOfAKind[type-3] != 0) {
                //checks in cases of a second Yahtzee
                if(type == 5) {
                    if(!yahtzeeBonus) {
                        yahtzeeBonus = true;
                        scoreOfAKind[type-3] += 100;
                        return scoreOfAKind[type-3];
                    }
                }
                //returns already stored value
                return scoreOfAKind[type-3];
            }
            //counts occurances of a die number and stores it in counts[]
            for (int i : dice) {
                counts[i - 1]++;
            }
            //iterates over counts[]
```

```
            for (int i : counts) {
                //if counts matches _ofAKind adds up all dice values and
stores it into scores
                if(i == 5 && type == 5) {
                    scoreOfAKind[type-3] = 50;
                    return scoreOfAKind[type-3];
                } else if (i >= type) {
                    for (int j : dice) {
                        score += j;
                    }
                    //interalizes scoreOfAKind[] with values from score,
and then returns the values
                    scoreOfAKind[type-3] = score;
                    return scoreOfAKind[type-3];
                }
            }
            //catches ArrayIndexOutOfBoundsException for 3 > type > 5
        } catch (ArrayIndexOutOfBoundsException g) {
            System.out.println("Error: " + g.getMessage());
        }
        //if there's no 3+ matching die numbers returns 0
        return 0;
    }

    /**
     * Calculates the score for a Full House category
     * Roll 3 dice with the same number, and 2 dice with the same number
     * @return The score for a Full House, which is 25 if the condition
is met, and 0 otherwise.
     */
    public int fullHouse() {
        int[] counts = new int[6];
        int[] dice = new
int[]{die1.value,die2.value,die3.value,die4.value,die5.value};
        //counts occurances of a die number and stores it in counts[]
        for (int i : dice) {
            counts[i - 1]++;
        }
        //interates over counts[] twice
        for(int j: counts) {
            for(int a: counts) {
                //initalizes fullHouse if conditions are met
                if(j == 3 && a == 2) {
                    fullHouse = 25;
                    return fullHouse;
                }
            }
        }
        //if conditions are not met return 0
        return 0;
    }

    /**
    * Calculates the score for a straight of four or five consecutive
numbers.
    * @param set the number of dice required for a straight, either 4 or
5
    * @return the score for a straight of the specified category, or 0 if
the dice do not form a straight of that category
```

```
    * @throws ArrayIndexOutOfBoundsException if the set value is not 4 or
5
    */
    public int straight(int set) {
        int[] dice = new
int[]{die1.value,die2.value,die3.value,die4.value,die5.value};
        Arrays.sort(dice);
        int notConsecCount = 0;
        try{
            if(set > 5 || set < 4) {
                throw new ArrayIndexOutOfBoundsException("Please enter 4-
5");
            }

            //if there's a stored value in the array stops the
calculations
            if (straight[set-4] != 0) {
                return straight[set-4];
            }

            //caculates nonconsective sequences
            for(int a: dice) {
                if(a+1 != a) {
                    notConsecCount++;
                }
            }

            //checks number of sequences
            if(notConsecCount == 1) {
                straight[0] = 30;
                return straight[0];
            } else if(notConsecCount == 0) {
                straight[1] = 40;
                return straight[1];
            }
            //catches ArrayIndexOutOfBoundsException for 4 > set > 5
        } catch (ArrayIndexOutOfBoundsException k) {
            System.out.println("Error: " + k.getMessage());
        }
        //if conditions are not met returns 0
        return 0;
    }

    public int Chance() {
        int[] dice = new
int[]{die1.value,die2.value,die3.value,die4.value,die5.value};
        //checks if Chance is already initalized
        if(Chance != 0) {
            return Chance;
        }
        for(int i: dice) {
            Chance += i;
        }
        return Chance;
    }

    /**
     * Returns for the score Chance
     *
     * @return the string for Chance
```

```
 */
public String getChance() {
    return("Chance: " + Chance);
}

/**
 * Returns the scores in scoreOfAKind
 *
 * @param "get" an integer that specifies the call for scoreOfAKind
 * @return the string for scoreOfAKind
 */
public String getScoreOfAKind(int get) {
    switch(get){
        case 3:
            return (get+"score of a kind: "+scoreOfAKind[0]);
        case 4:
            return (get+"score of a kind: "+scoreOfAKind[1]);
        case 5:
            return ("Yahtzee: "+scoreOfAKind[2]);
    }
    return ("0");
}

/**
 * Returns the score for Full House
 *
 * @return the string for the score Full House
 */
public String getFullHouse() {
    return("FullHouse: " + fullHouse);
}

/**
 * Returns the scores for straights
 *
 * @param "get" an integer that specifies the straight score
 * @return a string for straights scores
 */
public String getStraight(int get) {
    if(get == 4){
        return("Small Straight: " + straight[0]);
    } else {
        return("Large Straight: " + straight[1]);
    }
}

/**
 * Returns the scores from the Upper Section
 *
 * @param "get" an integer that specifies the Upper score
 * @return a string for Upper Section score
 */
public String getScoreUpper(int get) {
    return ("Score " + get + ": " + scoreUpper[get-1]);
}

/**
 * Returns the total scores in the Upper Section
 * Adds a bonus 35 score for a total upper score of over 63
 *
```

```
     * @return a string for the total score of the Upper Section
     */
    public String getUpperTotal() {
        int UpperTotal = scoreUpper[0] + scoreUpper[1]+ scoreUpper[2]+
            scoreUpper[3]+ scoreUpper[4]+ scoreUpper[5];
        this.upperTotal = UpperTotal;
        if(upperTotal > 63) {
            upperTotal += 35;
        }

        return ("Upper Total: "+upperTotal);
    }

    /**
     * Returns the total scores in the Lower Section
     *
     * @return a string for the total score of the Lower Section
     */
    public String getLowerTotal(){
        int LowerTotal = scoreOfAKind[0] + scoreOfAKind[1] +
scoreOfAKind[2] + fullHouse + straight[0] + straight[1] + Chance;
        this.lowerTotal = LowerTotal;
        return ("Lower Total: " +lowerTotal);
    }

    /**
     * Returns the Grand Total
     *
     * @return a string for the Grand Total
     */
    public String getGrandTotal() {
        grandTotal = upperTotal + lowerTotal;
        return ("Grand Total: " +grandTotal);
    }

}
```

```
/**
 * Abstracts one six-sided die
 *
 * @1.0.0
 */
public class Die6
{
  public int value;

  public Die6() {
    this.roll();
  }

  public int getValue() {
    return value;
  }

  public void roll() {
    this.value = (int)(Math.random() * 6) + 1;
  }

}
```