# Yahtzee Game

**Text.IO also used

**YahtzeeScore.java**

```
public class YahtzeeScore extends Category{
	public YahtzeeScore() {
		name = "Yahtzee";
	}
	public int calculateScore(int[] diceValues) {
		score = 0;
		int count= 0;
		boolean check = false;
		for(int i = 1; i < 7; i++) {
			count = 0;
			for(int j = 0; j < diceValues.length; j++) {
				if(diceValues[j] == i) {
					count++;
				}
			}
			if(count >= 5) {
				check = true;
				break;
			}
		}
		if (check == true) {
			score = 50;
		}
		return score;
	}
	public int getScore() {
		return score;
	}
	public boolean isUsed() {
		return used;
	}
	public void setUsed(){
		used = true;
	}
	public String getName() {
		return name;
	}
}
```

**YahtzeeRunner.java**

```
public class YahtzeeRunner {
	public static void main(String[] args) {
		System.out.println("**Starting Game**");
		System.out.println("You have 3 turns per round");
		YahtzeeGame game = new YahtzeeGame();
		ScoreBoard scoreBoard = new ScoreBoard();

		for(int i = 0; i < 13; i++) {
			game.getDice().rollAll();
			while(game.getTurn() >= 1) {

System.out.println("Rolling...");

System.out.println(game.getDice());
				System.out.println("You have " + game.changeTurns() + " turns left in this round");

				if(game.getTurn() > 0) {

System.out.println("Would you like to roll again? (y) for yes, (n) for no");
					if (TextIO.getln().equals("n")) {
						break;
					}

System.out.println("Which die would you like to hold (y) for yes and (n) to no (ex: respond yyynn to hold the first 3)");
					String hold = TextIO.getln();
			game.getDice().rollSome(hold);
				}

			}
			System.out.println("For this round, your scores read:");

scoreBoard.setSomeScores(game.getDice().getDiceValues());

scoreBoard.printScoreBoardPreview();
			System.out.println("Which score would you like to pick? (enter a nummber)");
			String answer = TextIO.getln();
			int answerInt = Integer.parseInt(answer);
			scoreBoard.setUsedScore(answerInt);
			System.out.println("Your actual score board reads:");
			scoreBoard.printActualScoreBoard();

			game.resetTurns(); //makes it so that we can start a new game
		}
```

**YahtzeeDice.java**

```
public class YahtzeeDice{
	private Die[] dice = new Die[5];

//Constructors
	public YahtzeeDice() {
		for(int i = 0; i < dice.length; i++) {
			dice[i] = new Die();
		}
	}

	public YahtzeeDice(int numSides) {
		for(int i = 0; i < dice.length; i++) {
			dice[i] = new Die(numSides);
		}
	}

//Instance Methods
	public int roll() {
		int sum = 0;
		for(int i = 0; i < dice.length; i++) {
			sum += dice[i].getCurrentValue();
		}
		return sum;
	}

	public void rollAll() {
		for(int i = 0; i < 5; i++) {
			dice[i].roll();
		}
	}
	public void rollSome(String str) {

		for(int i = 0; i < str.length(); i++) {
			if (str.charAt(i) == 'n') {
				dice[i].roll();
			} else {
				dice[i] = dice[i];
			}
		}
	}

	public Die[] getDice() {
		return dice;
	}

	public int[] getDiceValues() {
		int[] values = new int[5];
		for(int i = 0; i < values.length; i++) {
			values[i] =
dice[i].getCurrentValue();
		}
		return values;
	}

	public String toString() {
		String yahtzee = "The 5 dice read: ";
	}
}
```

**YahtzeeGame.java**

```
public class YahtzeeGame{
	private YahtzeeDice gameDice = new YahtzeeDice();
	private int rounds = 13;
	private int turnsPerRound = 3;

	public int rollGameDice() {
		return gameDice.roll(); //gets the sum
	}

	public YahtzeeDice getDice() {
		return gameDice;
	}


	public int changeTurns() {
		turnsPerRound--;
		return turnsPerRound;
	}

	public int getTurn() {
		return turnsPerRound;
	}

	public void resetTurns() {
		turnsPerRound = 3;
	}
}
```

**Category.java**

```
public abstract class Category {
	protected int score;
	protected String name;
	protected boolean used = false;


	public abstract int calculateScore(int[]
diceValues);
	public abstract boolean isUsed();
	public abstract int getScore();
	public abstract void setUsed();
	public abstract String getName();

}
```

**Chance.java**

```
public class Chance extends Category{

	public Chance() {
		name = "Chance";
	}

	public int calculateScore(int[] diceValues) {
		score = 0;
		for(int i = 0; i < diceValues.length; i++)
```

```
		for(int i = 0; i < dice.length; i++) {
			if(i < dice.length - 1)
				yahtzee +=
dice[i].getCurrentValue() + ", ";
			else
				yahtzee +=
dice[i].getCurrentValue() + ".";
		}
		return yahtzee;

		/*return "Each dice has " +
dice[0].getNumSides() + " sides.  The values of the 5
dice are " +

*java.util.Arrays.toString(getDiceValues()) + ".  The sum
of the all the dice is " + roll();
		*I use dice[0] to get the number of sides
for each of the die, each die has the
		*same number of sides.  This also makes it
so that I don't repeat the number of
		*sides each time.  This will never be null
because it will automatically initiate
		*to 6 when the die is created, if not
specified otherwise.
		*/
	}
}
```

```
{
			score += diceValues[i];
		}
		return score;
	}

	public int getScore() {
		return score;
	}

	public boolean isUsed() {
		return used;
	}

	public void setUsed(){
		used = true;
	}

	public String getName() {
		return name;
	}

}
```

**ScoreBoard.java**

```
public class ScoreBoard {
	private Category[] scores;
	public ScoreBoard() {
		scores = new Category[13];
		scores[0] = new OneScore();
		scores[1] = new TwoScore();
		scores[2] = new ThreeScore();
		scores[3] = new FourScore();
		scores[4] = new FiveScore();
		scores[5] = new SixScore();
		scores[6] = new ThreeOfKind();
		scores[7] = new FourOfKind();
		scores[8] = new LargeStraight();
		scores[9] = new SmallStraight();
		scores[10] = new FullHouse();
		scores[11] = new YahtzeeScore();
		scores[12] = new Chance();
	}

	public void setSomeScores(int[] diceValues) {
		for(int i = 0; i < 13; i++) {
			if(!scores[i].isUsed()) {

scores[i].calculateScore(diceValues);
			}
		}
	}
	public void setAllScores(int[] diceValues) {
		for(int i = 0; i < 13; i++) {
```

**OneScore.java**

```
public class OneScore extends Category{
	public OneScore() {
		name = "Ones";
	}
	public int calculateScore(int[] diceValues) {
		score = 0;
		for (int i = 0; i < diceValues.length;
i++) {
			if (diceValues[i] == 1) {
				score++;
			}
		}
		return score;
	}
	public int getScore() {
		return score;
	}
	public boolean isUsed() {
		return used;
	}
	public void setUsed(){
		used = true;
	}
	public String getName() {
		return name;
	}
}
```

**TwoScore.java**

```
public class TwoScore extends Category{
```

```
scores[i].calculateScore(diceValues);
                }
        }
        public void printScoreBoardPreview() {
                for(int i = 0; i < 13; i++) {
                        if(!scores[i].isUsed()) {
                                System.out.println(i + ". "
+ scores[i].getName() + ": " + scores[i].getScore());
                        }
                }
        }

        public void printActualScoreBoard() {
                for(int i = 0; i < 13; i++) {
                        if(scores[i].isUsed()) {
                                System.out.println(i + ". "
+ scores[i].getName() + ": " + scores[i].getScore());
                        } else {
                                System.out.println(i + ". "
+ scores[i].getName() + ": 0");
                        }
                }
        }

        public void setUsedScore(int i) {
                scores[i].setUsed();
        }
// do if answer = i, scores i - 1.setUsed
}
```

**Die.java**

```
import java.lang.Math;
public class Die{
        //instance variables - attributes
        private int numSides;
        private int currentValue;

        //constructor
        public Die() {
                numSides = 6;
                roll(); //sets current value to random
number
        }

        public Die(int sides) {
                numSides = sides;
                roll();
        }

        public Die(int sides, int value) {
                numSides = sides;
                currentValue = value;
        }

        //instance methods - behaviors
        public int roll(){
                currentValue =
(int)(Math.random()*numSides) + 1;
```

```
        public TwoScore() {
                name = "Twos";
        }
        public int calculateScore(int[] diceValues) {
                score = 0;
                for (int i = 0; i < diceValues.length;
i++) {
                        if (diceValues[i] == 2) {
                                score += 2;
                        }
                }
                return score;
        }

        public int getScore() {
                return score;
        }

        public boolean isUsed() {
                return used;
        }

        public void setUsed(){
                used = true;
        }

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

**ThreeScore.java**

```
public class ThreeScore extends Category{

        public ThreeScore() {
                name = "Threes";
        }

        public int calculateScore(int[] diceValues) {
                score = 0;
                for (int i = 0; i < diceValues.length;
i++) {
                        if (diceValues[i] == 3) {
                                score += 3;
                        }
                }
                return score;
        }

        public int getScore() {
                return score;
        }

        public boolean isUsed() {
                return used;
        }

        public void setUsed(){
                used = true;
```

```
                return currentValue;
        }

        public String toString() {
                return "Num sides: " + numSides + ",
Value: " + currentValue;
        }

        public int getCurrentValue() {
                return currentValue;
        }

        public int getNumSides() {
                return numSides;
        }

}
```

**ThreeOfKind.java**

```
public class ThreeOfKind extends Category{
        public ThreeOfKind() {
                name = "Three of a Kind";
        }
        public int calculateScore(int[] diceValues) {
                score = 0;
                int count= 0;
                boolean check = false;
                for(int i = 1; i < 7; i++) {
                        count = 0;
                        for(int j = 0; j <
diceValues.length; j++) {
                                if(diceValues[j] == i) {
                                        count++;
                                }
                        }
                        if(count >= 3) {
                                check = true;
                                break;
                        }
                }
                if (check == true) {
                        for(int i = 0; i <
diceValues.length; i++) {
                                score += diceValues[i];
                        }
                }
                return score;
        }

        public int getScore() {
                return score;
        }

        public boolean isUsed() {
                return used;
        }

        public void setUsed(){
                used = true;
        }

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

**FourScore.java**

```
public class FourScore extends Category{

        public FourScore() {
                name = "Fours";
        }

        public int calculateScore(int[] diceValues) {
                score = 0;
                for (int i = 0; i < diceValues.length;
i++) {
                        if (diceValues[i] == 4) {
                                score += 4;
                        }
                }
                return score;
        }

        public int getScore() {
                return score;
        }

        public boolean isUsed() {
                return used;
        }

        public void setUsed(){
                used = true;
        }

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

**FiveScore.java**

```
public class FiveScore extends Category{

        public FiveScore() {
                name = "Fives";
        }
        public int calculateScore(int[] diceValues) {
                score = 0;
                for (int i = 0; i < diceValues.length;
i++) {
                        if (diceValues[i] == 5) {
                                score += 5;
                        }
                }
                return score;
        }

        public int getScore() {
```

```
	}

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

**FourOfKind.java**

```
public class FourOfKind extends Category{

	public FourOfKind() {
		name = "Four of a Kind";
	}

	public int calculateScore(int[] diceValues) {
		score = 0;
		int count= 0;
		boolean check = false;
		for(int i = 1; i < 7; i++) {
			count = 0;
			for(int j = 0; j < 
diceValues.length; j++) {
				if(diceValues[j] == i) {
					count++;
				}
			}
			if(count >= 4) {
				check = true;
				break;
			}
		}
		if (check == true) {
			for(int i = 0; i < 
diceValues.length; i++) {
				score += diceValues[i];
			}
		}
		return score;
	}

	public int getScore() {
		return score;
	}

	public boolean isUsed() {
		return used;
	}

	public void setUsed(){
		used = true;
	}
	public String getName() {
		return name;
	}
}
```

**FullHouse.java**

```
import java.util.Arrays;
public class FullHouse extends Category{

	public FullHouse() {
		return score;
	}

	public boolean isUsed() {
		return used;
	}

	public void setUsed(){
		used = true;
	}

	public String getName() {
		return name;
	}
}

public class SixScore extends Category{

	public SixScore() {
		name = "Sixes";
	}

	public int calculateScore(int[] diceValues) {
		score = 0;
		for (int i = 0; i < diceValues.length; 
i++) {
			if (diceValues[i] == 6) {
				score += 6;
			}
		}
		return score;
	}

	public int getScore() {
		return score;
	}

	public boolean isUsed() {
		return used;
	}

	public void setUsed(){
		used = true;
	}

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

**SmallStraight.java**

```
import java.util.Arrays;
public class SmallStraight extends Category{

	public SmallStraight() {
		name = "Small Straight";
	}

	public int calculateScore(int[] diceValues) {
		score = 0;
```

```
                name = "Full House";
        }
        public int calculateScore(int[] diceValues) {
                score = 0;
                Arrays.sort(diceValues);
                if(diceValues[0] == diceValues[1] &&
diceValues[0] == diceValues[2] && diceValues[3] ==
diceValues[4]) {
                        score = 25;
                }
                if(diceValues[0] == diceValues[1] &&
diceValues[2] == diceValues[3] && diceValues[2] ==
diceValues[4]) {
                        score = 25;
                }
                return score;
        }

        public int getScore() {
                return score;
        }

        public boolean isUsed() {
                return used;
        }

        public void setUsed(){
                used = true;
        }
        public String getName() {
                return name;
        }

}
```

```
                Arrays.sort(diceValues);
                int[] temp = new int[5];
                int j = 0;
                for(int i = 0; i < diceValues.length - 1;
i++) {
                        if(diceValues[i] != diceValues[i +
1]) {
                                temp[j] = diceValues[i];
                        } else {
                                temp[j] = 0;
                        }
                        j++;
                }
                temp[4] = diceValues[4];
                diceValues = temp;
                Arrays.sort(temp);
                if(diceValues[1] == 3 && diceValues[2] ==
4 && diceValues[3] == 5 &&
                        diceValues[4] == 6) {
                        score = 30;
                }
                if(diceValues[1] == 2 && diceValues[2] ==
3 && diceValues[3] == 4 &&
                        diceValues[4] == 5) {
                        score = 30;
                }
                if(diceValues[1] == 1 && diceValues[2] ==
2 && diceValues[3] == 3 &&
                        diceValues[4] == 4) {
                        score = 30;
                }
                if(diceValues[0] == 1 && diceValues[1] ==
2 && diceValues[2] == 3 &&
                        diceValues[3] == 4) {
                        score = 30;
                }
                if(diceValues[0] == 1 && diceValues[1] ==
2 && diceValues[2] == 3 && diceValues[3] == 4 &&
                        diceValues[4] == 5) {
                        score = 30;
                }
                if(diceValues[0] == 2 && diceValues[1] ==
3 && diceValues[2] == 4 && diceValues[3] == 5 &&
                        diceValues[4] == 6) {
                        score = 30;
                }
                return score;
        }

        public int getScore() {
                return score;
        }

        public boolean isUsed() {
                return used;
        }

        public void setUsed(){
                used = true;
        }
```

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

**LargeStraight.java**

```
import java.util.Arrays;
public class LargeStraight extends Category{

	public LargeStraight() {
		name = "Large Straight";
	}
	public int calculateScore(int[] diceValues) {
		score = 0;
		Arrays.sort(diceValues);
		if(diceValues[0] == 1 && diceValues[1] ==
2 && diceValues[2] == 3 && diceValues[3] == 4 &&
			diceValues[4] == 5) {
			score = 40;
		}
		if(diceValues[0] == 2 && diceValues[1] ==
3 && diceValues[2] == 4 && diceValues[3] == 5 &&
			diceValues[4] == 6) {
			score = 40;
		}
		return score;
	}

	public int getScore() {
		return score;
	}

	public boolean isUsed() {
		return used;
	}

	public void setUsed(){
		used = true;
	}

	public String getName() {
		return name;
	}

}
```