

Dice Game: Build Your Own Classes

Name: _____

Due: _____

You will write a small dice game from scratch, using **four classes in four files**: Die, Player, Game, and DiceGame. There is no starter code. Every design decision inside each class is yours, as long as the class behaves as described below.

The game

- A game has some number of players. Each player has their own **three dice**.
- In each round, every player rolls all three of their dice. The player with the **highest total** wins the round and gets one point.
- If two or more players tie for the highest total, **nobody** gets a point that round.
- After all the rounds are played, the player with the most points is the winner.

The classes

Each class below lists what it must be able to do. You decide its instance variables, and you may add private helper methods. Use the exact public method names given so that your classes work with anyone else's.

Die

- A Die has a number of sides, which never changes once the die is created.
- A static defaultSides value, which starts at 6 and is shared by every die. `public static void setDefaultSides(int s)` changes it. Ignore values less than 1.
- `Die()` creates a die with the current default number of sides.
- `Die(int sides)` creates a die with that many sides. If sides is less than 1, print a message and use the default instead.
- `public int roll()` returns a random whole number from 1 to the number of sides, inclusive.

Player

- A Player has a name, their own three dice, and a point total that starts at 0.
- `Player(String name)` creates the player and their dice.
- `public int takeTurn()` rolls all three dice, prints a line like `Player 2 rolls 13`, and returns the total.
- `public void addPoint()` adds one point.
- `public int getTotalPoints()` returns the point total.
- `public String toString()` returns something like `Player 2 has 5 points`.

Game

- A Game keeps its players in an **array** of Player objects.
- `Game(int numPlayers)` creates that many players, named `Player 1`, `Player 2`, and so on.
- `public void playRound()` plays one round as described above and prints who won it, or that there was no winner.
- `public void play(int rounds)` plays that many rounds.
- `public Player getWinner()` returns the Player **object** with the most points. If players tie for the most, you may return any one of them.

DiceGame

This class has only `public static void main(String[] args)`. It must do these three things, in this order:

1. Create a `Die` using `Die(int sides)` and roll it to test it. (Hint: what does a 1-sided die always roll?)
2. Call `Die.setDefaultSides` to choose how many sides the game's dice will have.
3. Create a `Game`, play at least 10 rounds, and print the winner.

Sample output (3 players, 8-sided dice, 3 rounds)

```
Test die rolls 1

Playing round
Player 1 rolls 12
Player 2 rolls 11
Player 3 rolls 16
Winner: Player 3 has 1 points

Playing round
Player 1 rolls 16
Player 2 rolls 13
Player 3 rolls 17
Winner: Player 3 has 2 points

Playing round
Player 1 rolls 16
Player 2 rolls 14
Player 3 rolls 13
Winner: Player 1 has 1 points
Player 3 has 2 points
```

Your numbers will be different. The format doesn't have to match exactly.

Checklist

- ☐ Four classes, each in its own file.
- ☐ Every instance variable is `private`.
- ☐ Static members are used only through the class name, as in `Die.setDefaultSides(8)`.
- ☐ `Game` stores its players in an array, and each `Player` stores their dice in an array.
- ☐ `main` uses both `Die(int sides)` and `Die.setDefaultSides`.

Extension

Handle ties for the overall winner. If two or more players finish with the most points, the program should report all of them instead of picking one. You decide how: change what `getWinner` returns, add a new method, or something else. Explain your choice in a comment.