159 XTENDED

A WEEKLY REVIEW

EXCEPTION HANDLING

Pull up **Socrative.com**

and join room **XTEND159**

WHAT ARE EXCEPTIONS?

WHEN DO THEY OCCUR?

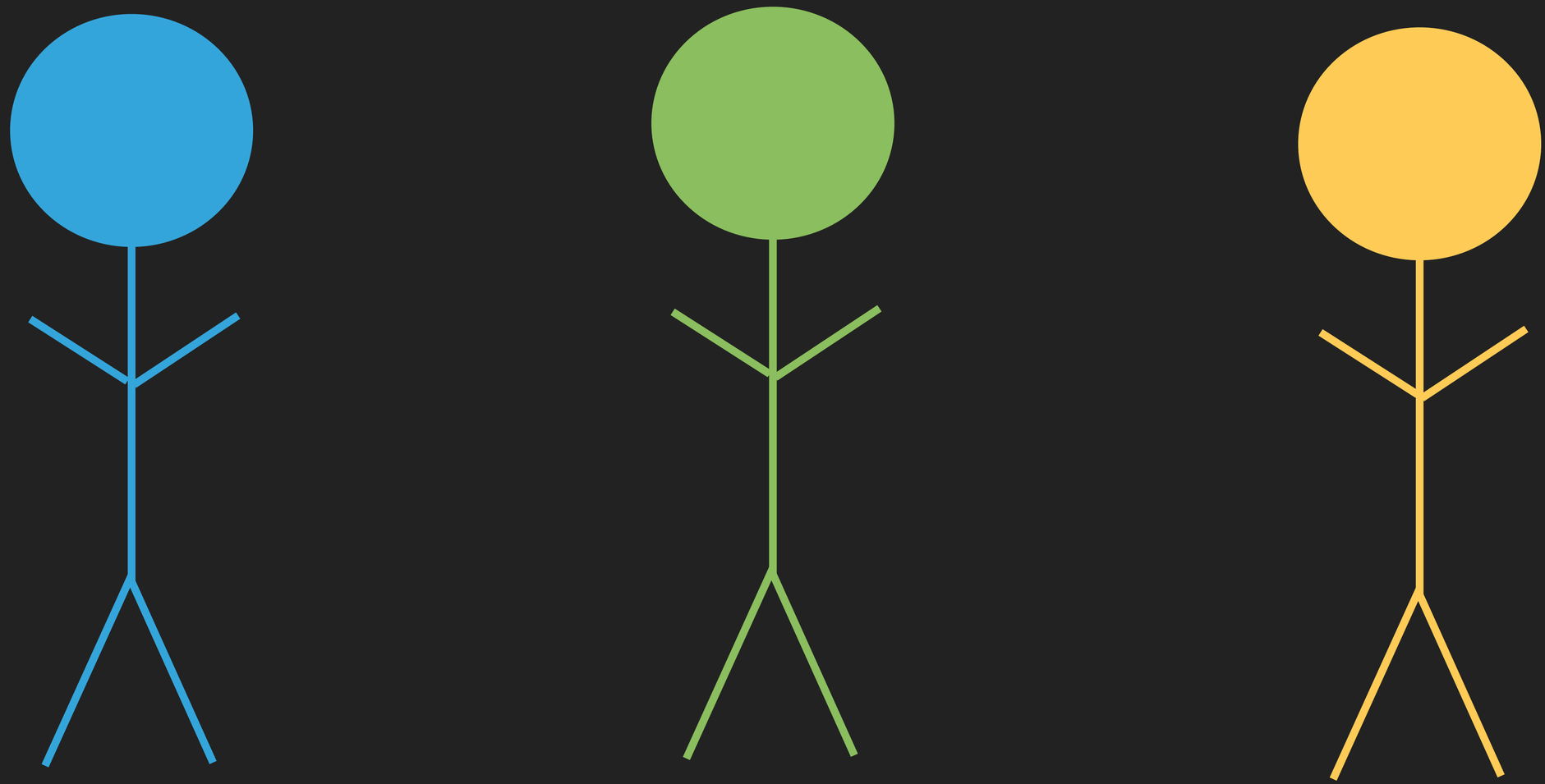
HOW DO WE HANDLE THEM?

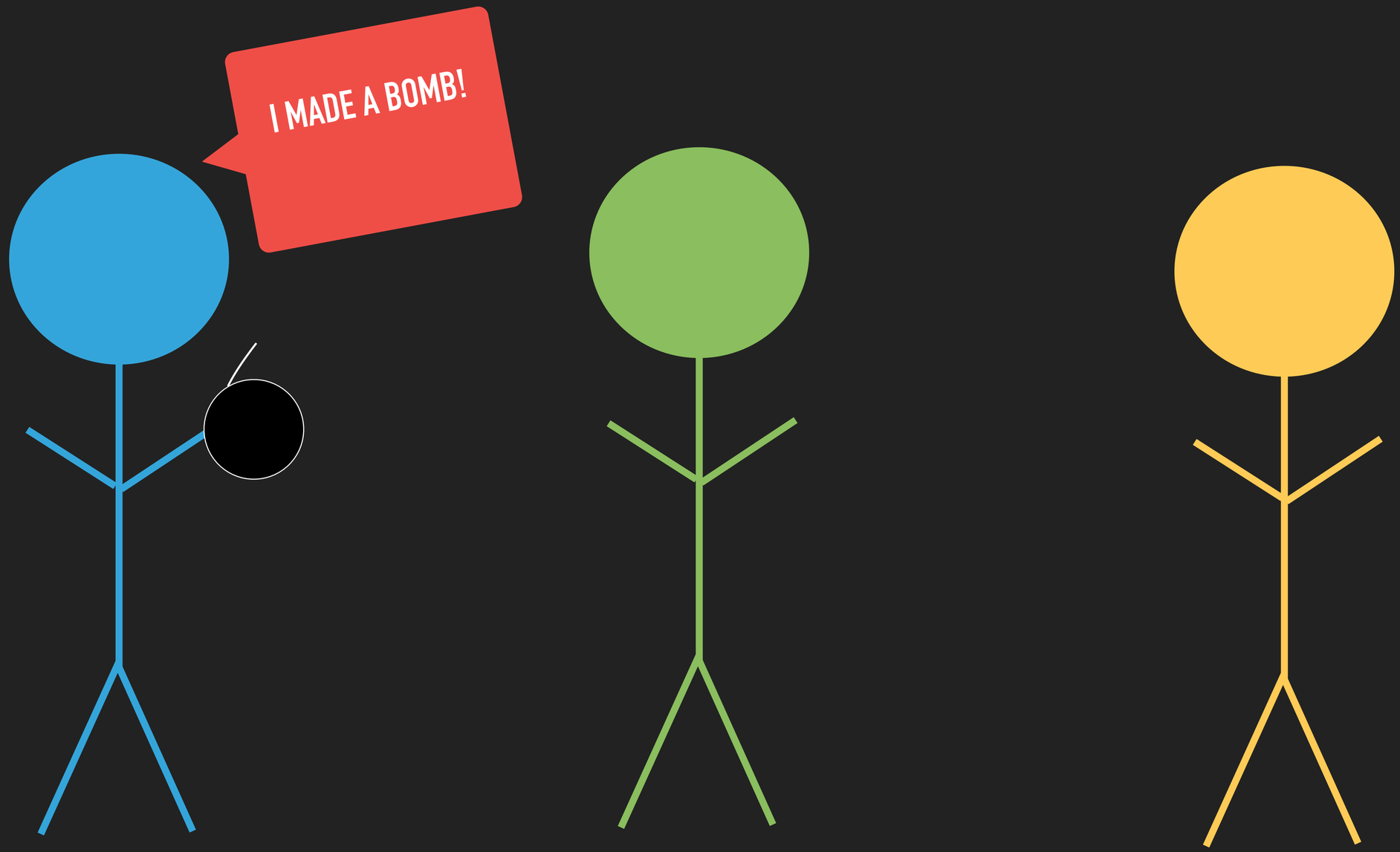
WHAT ARE EXCEPTIONS?

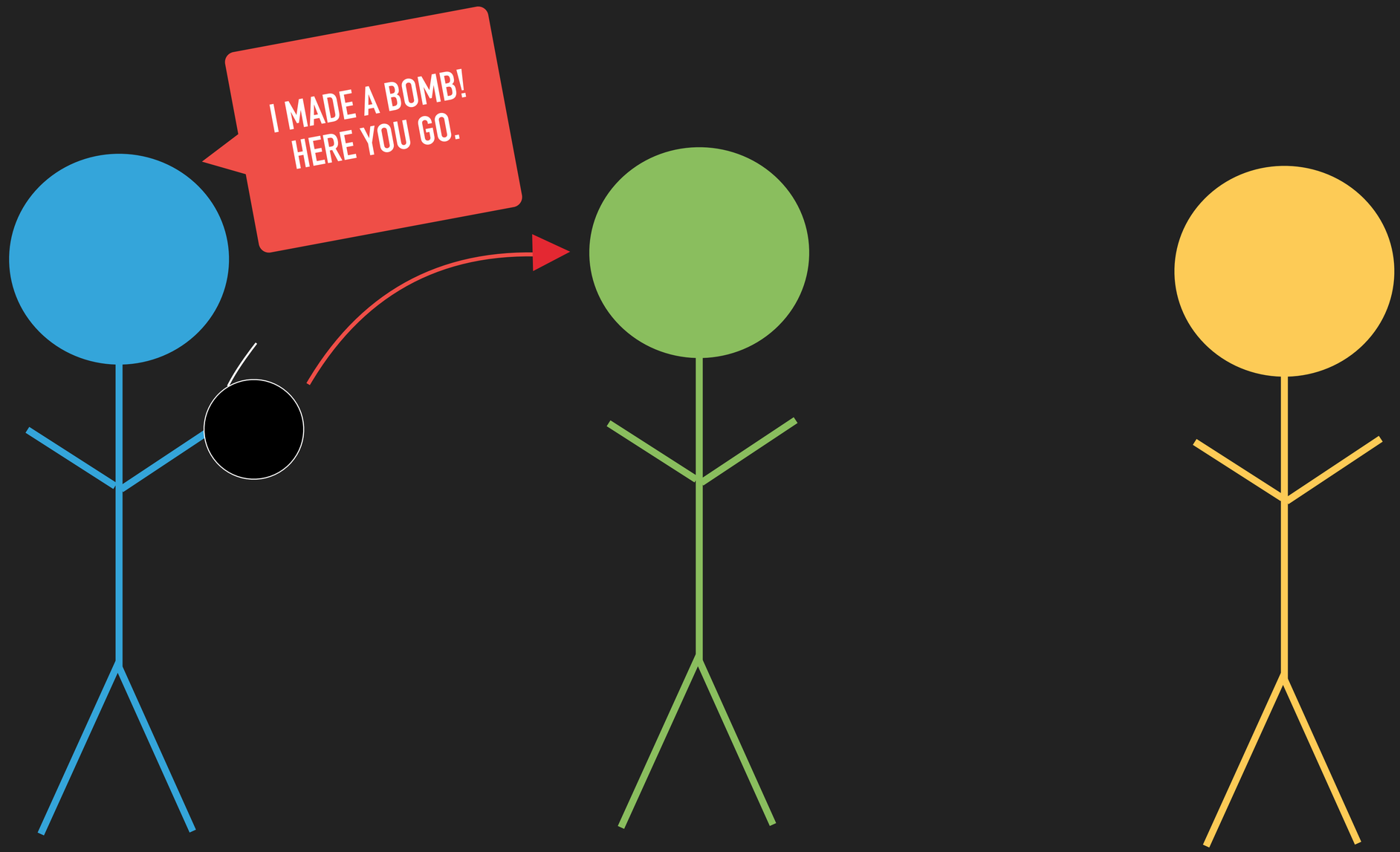
- Exceptions are events that disrupt the normal flow of program execution.
  - Translated: When something abnormal happens in our program, exceptions occur.
- They tell us…
  - What went wrong?
  - Where did something go wrong?

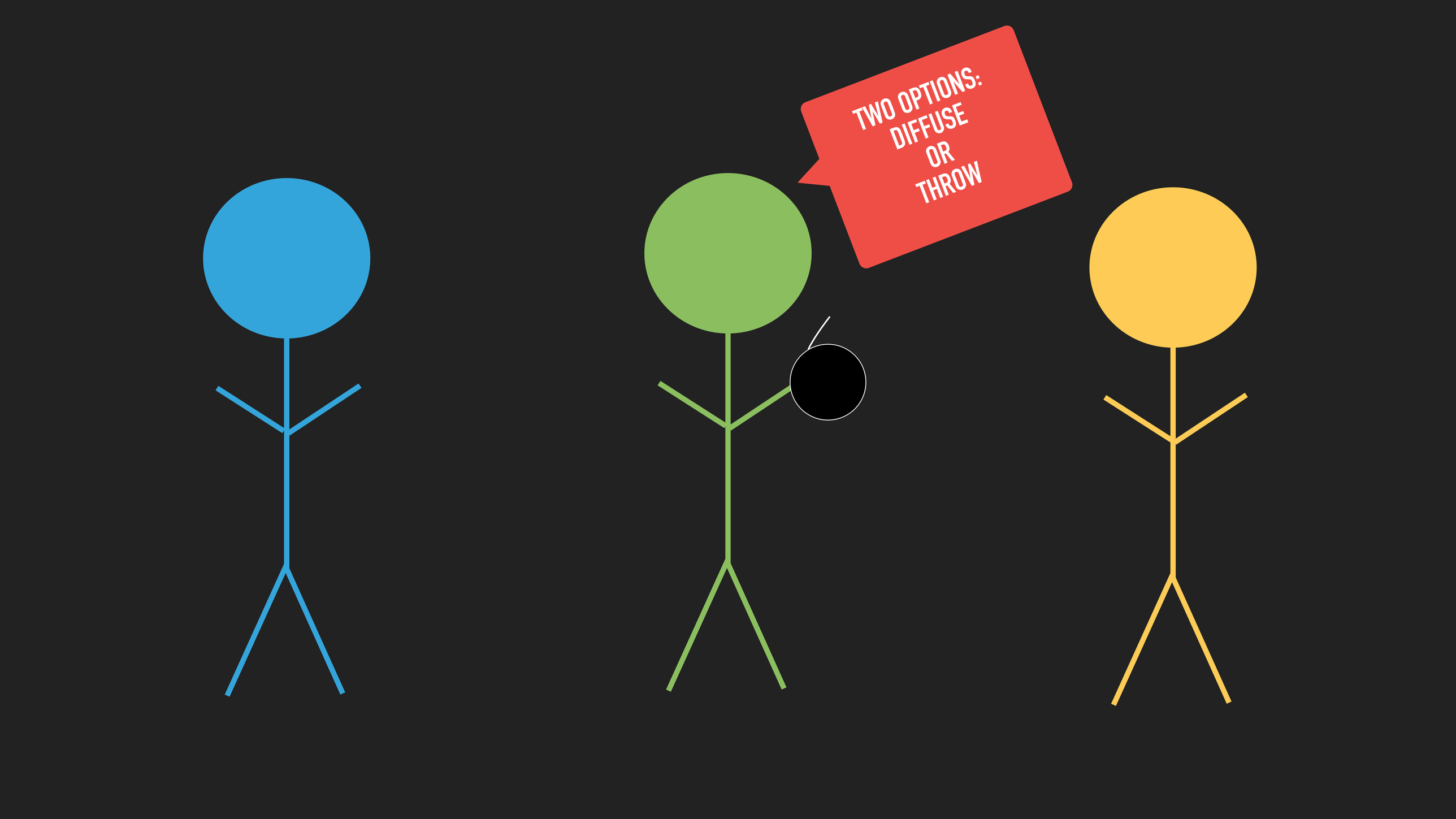
WHEN DO THEY OCCUR?

- When a method can't handle the input it's been given.
- When there is no appropriate return value to indicate an error, we can throw an exception.
- Examples…
  - Dividing by zero results in an ArithmeticException
  - Accessing an invalid array index results in an IndexOutOfBoundsException



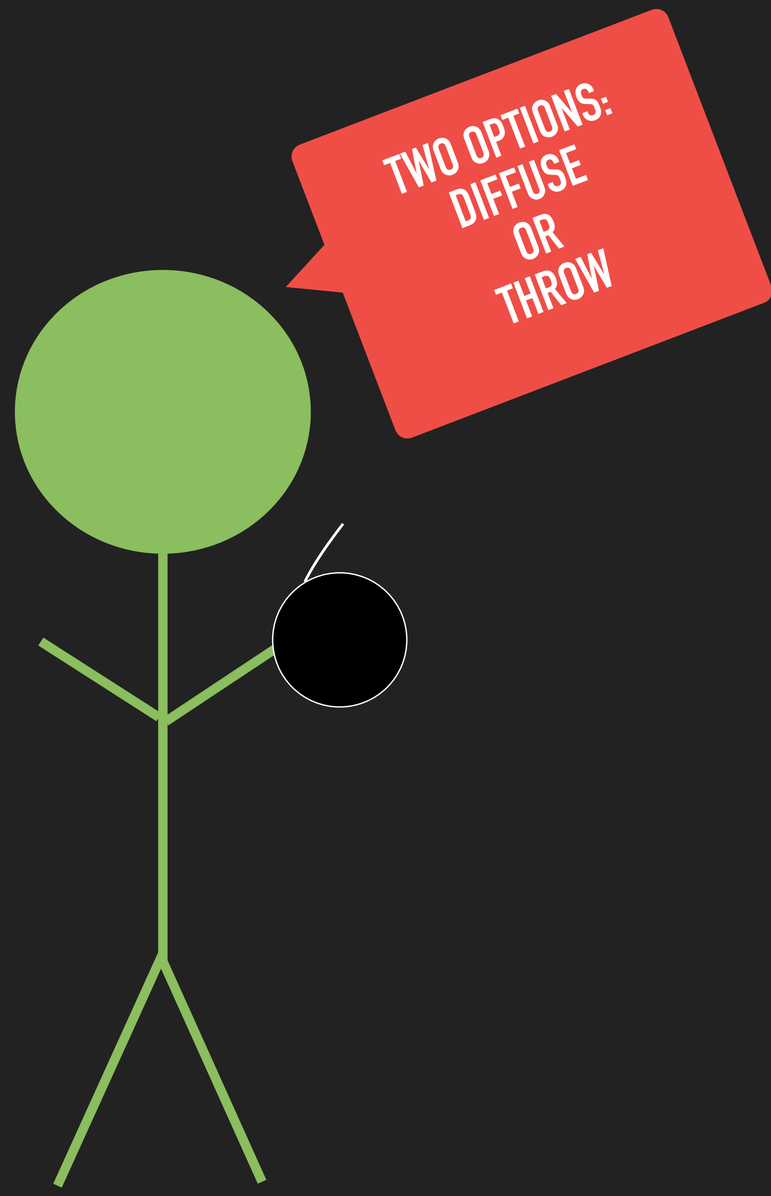

I MADE A BOMB!


I MADE A BOMB!
HERE YOU GO.


TWO OPTIONS:
DIFFUSE
OR
THROW

DIFFUSE OR THROW?

- I can diffuse the bomb if I know how.
- If I don't know how to diffuse it,
  I can throw it to the next person.
- If I throw it to the next person,
  they had better know how to diffuse it.
- If no one defuses the bomb,
  it will explode.



EXCEPTION HANDLING ACTIVITY

- Consider the following method:

```
public static void divideByTwo()
{
    Scanner in;
    int x;
    String line;

    in = new Scanner(System.in);
    System.out.println("Enter an integer: ");
    line = in.nextLine();
    x = Integer.parseInt(line);
    System.out.println("Half of x is " + x/2);
}
```

WHERE MIGHT AN EXCEPTION OCCUR?

EXCEPTION HANDLING ACTIVITY

- Consider the following method:

```
public static void divideByTwo()
{
    Scanner in;
    int x;
    String line;

    in = new Scanner(System.in);
    System.out.println("Enter an integer: ");
    line = in.nextLine();
    x = Integer.parseInt(line);
    System.out.println("Half of x is " + x/2);
}
```

WHAT HAPPENS IF THE USER INPUTS THE STRING "GO DUKES"?

THE PARSEINT METHOD WILL THROW AN EXCEPTION

EXCEPTION HANDLING ACTIVITY

- Consider the following method:

```
public static void divideByTwo() throws NumberFormatException
{
    Scanner in;
    int x;
    String line;

    in = new Scanner(System.in);
    System.out.println("Enter an integer: ");
    line = in.nextLine();
    x = Integer.parseInt(line);
    System.out.println("Half of x is " + x/2);
}
```

IF WE THROW…

EXCEPTION HANDLING ACTIVITY

- Consider the following method:

```
public static void divideByTwo(){

    // Some Code redacted

    try
    {
        x = Integer.parseInt(line);
    }
    catch (NumberFormatException e)
    {
        System.out.println("Invalid input.");
    }

    System.out.println("Half of x is " + x/2);
}
```

IF WE TRY/CATCH…

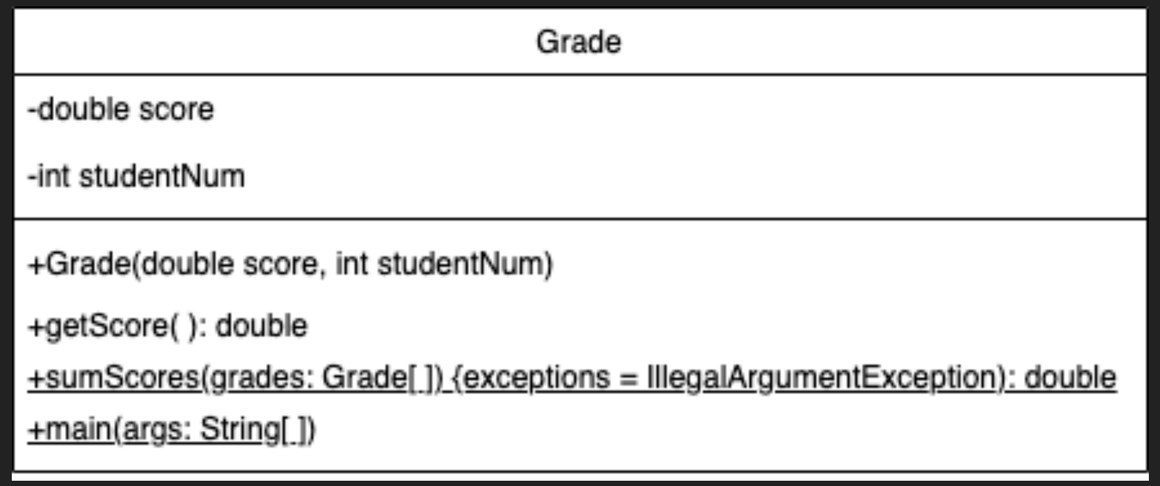
CHECKED AND UNCHECKED EXCEPTIONS

- Checked exceptions must be handled in a Try/Catch block or by throwing.
  - The compiler checks to ensure that they are handled (hence, they're checked).
  - These Exceptions extend RuntimeException.
- Unchecked exceptions may occur, but they don't have to be handled.
  - Examples include IndexOutOfBounds and ArithmeticException
  - These Exceptions extend Exception.

EXCEPTIONS ACTIVITY

- Consider the following UML diagram:

  Note: sumScores should return a double.



- Implement the class without the two static methods.
- Are Grade object mutable or immutable?
- Implement the `sumScores` method. The method should throw an `IllegalArgumentException` if `grades` is null or empty.
- Implement a `main` method in the class. The method should create an array with two grades and call `sumScores` with it.

EXCEPTIONS ACTIVITY

*does not include main

```
public class Grade
{
    // Attributes
    private double score;
    private int studentNum;

    public Grade(double score, int studentNum)
    {
        this.score = score;
        this.studentNum = studentNum;
    }

    public double getScore()
    {
        return this.score;
    }

    public static double sumScores(Grade[] grades) throws IllegalArgumentException
    {
        if (grades == null || grades.length == 0)
            throw new IllegalArgumentException();

        double total = 0.0;

        for (int i = 0; i < grades.length; i++)
        {
            total += grades[i].getScore();
        }

        return total;
    }
}
```

EXCEPTIONS ACTIVITY

```
public class Grade
{
    // Code omitted

    public static void main(String[] args)
    {
        double totalScore = 0.0;

        Grade[] grades = new Grade[2];
        grades[1] = new Grade(90.0, 10011001);
        grades[2] = new Grade(50.0, 99999999);

        try
        {
            totalScore = Grade.sumScores(grades);
        }
        catch (IllegalArgumentException e)
        {
            System.out.println(“Invalid grade list.”);
        }
    }
}
```

The main method attempts to call sumScores on the created Grades array, which throws an IllegalArgumentException if our array is null or empty. In the case of a null or empty array, our method will print “Invalid grade list.”