Exceptions

When you first started programming, getting an exception meant you made a mistake. But not all exceptions are run-time errors. Exceptions can be useful for handling special cases.

Manager: Recorder:

Presenter: Reflector:

Content Learning Objectives

After completing this activity, students should be able to:

- Throw run-time exceptions with specific detail messages.
- Use `try` and `catch` blocks to handle run-time exceptions.
- Discuss when throwing and catching exceptions is useful.

Process Skill Goals

During the activity, students should make progress toward:

- Identifying methods and line numbers in a stack trace. (Information Processing)

Model 1 Illegal Arguments

Consider a Java class that represents a playing card. Each card object has two values: `int rank` and `int suit`. Rank values range from 1 to 13, where 1=Ace, 11=Jack, 12=Queen, and 13=King. Suit values range from 0 to 3, where 0=Clubs, 1=Diamonds, 2=Hearts, and 3=Spades. The `Card` constructor *throws an exception* if either value is out of range:

```
public Card(int rank, int suit) {
    if (rank < 1 || rank > 13) {
        throw new IllegalArgumentException(
                    "invalid rank: " + rank);
    }
    if (suit < 0 || suit > 3) {
        throw new IllegalArgumentException(
                    "invalid suit: " + suit);
    }
    this.rank = rank;
    this.suit = suit;
}
```

Questions (20 min) Start time:

1. What arguments are needed to construct the five cards shown?

a) 10 of Hearts: `new Card( , )`

b) Jack of Hearts: `new Card( , )`

c) Queen of Hearts: `new Card( , )`

d) King of Hearts: `new Card( , )`

e) Ace of Hearts: `new Card( , )`

2. Give examples of illegal arguments for the `Card` constructor:

a) Rank too low: `new Card( , )`

b) Rank too high: `new Card( , )`

c) Suit too low: `new Card( , )`

d) Suit too high: `new Card( , )`

3. Type one of your examples with an **incorrect suit** into *Test.java*. What is the result when you run the program? (Don't just say "error"—describe the error message in detail.)

4. Remove (or comment out) all but the last two lines of the constructor in *Card.java*. What is the result of the `Test` program now?

5. Look carefully at the error message from the previous question. List all methods that were called, in the order they were called:

Class Name	Method Name	Line Number

6. On what line number of what file did the index out of bounds exception occur?

7. Was there anything wrong with that particular line of code? Why or why not?

8. Give two reasons why the exception in #3 is more useful than the exception in #4.

9. Restore (or uncomment) the original code for the `Card` constructor that checks the rank and suit values. Then modify the `parseRank` and `parseSuit` methods to throw an illegal argument exception instead of returning `-1` if the rank or suit is not found. Write your new code below.

a) Last line of `parseRank`:

b) Last line of `parseSuit`:

Model 2 Catching Exceptions

If you know that an exception might occur, you can `try` to run the code anyway, `catch` the exception if it occurs, and run other code to deal with it. Consider the the following example from *User.java*, a program that prompts the user to input a rank and a suit:

```
try {
    int rankInt = Integer.parseInt(rankStr);
    int suitInt = Integer.parseInt(suitStr);
    card = new Card(rankInt, suitInt);
} catch (NumberFormatException exc) {
    card = new Card(rankStr, suitStr);
}
System.out.println("Your card is: " + card);
```

Questions (20 min) Start time:

10. Which version of the `Card` constructor is called . . .

a) in the `try` block?

b) in the `catch` block?

11. Run *User.java* with each of the inputs below, and record the output.

Enter a rank:	Enter a suit:	Your card is:
1	2	
Ace	Hearts	
ace	hearts	

12. In the previous question, which inputs caused the `NumberFormatException`? (Note: The `NumberFormatException` is not shown when the program runs.)

13. Explain in your own words what the `try` and `catch` blocks do in this program.

14. Run the program again, and input `-1` for both the rank and the suit. What is the result?

15. Add a second `catch` block (just before the last `println` statement) to handle the exception from the previous question. The `catch` block should print `exc.getMessage()`. Write your new code below.

16. Describe in your own words how you could modify the current code to keep asking the user for input until they enter a valid rank and suit.

17. Implement your idea from the previous question. Test your program, and make sure it works correctly. Then write the first and last four lines of your code below.

First four lines:

Last four lines:

18. Summarize the two uses of exceptions in today's activity.