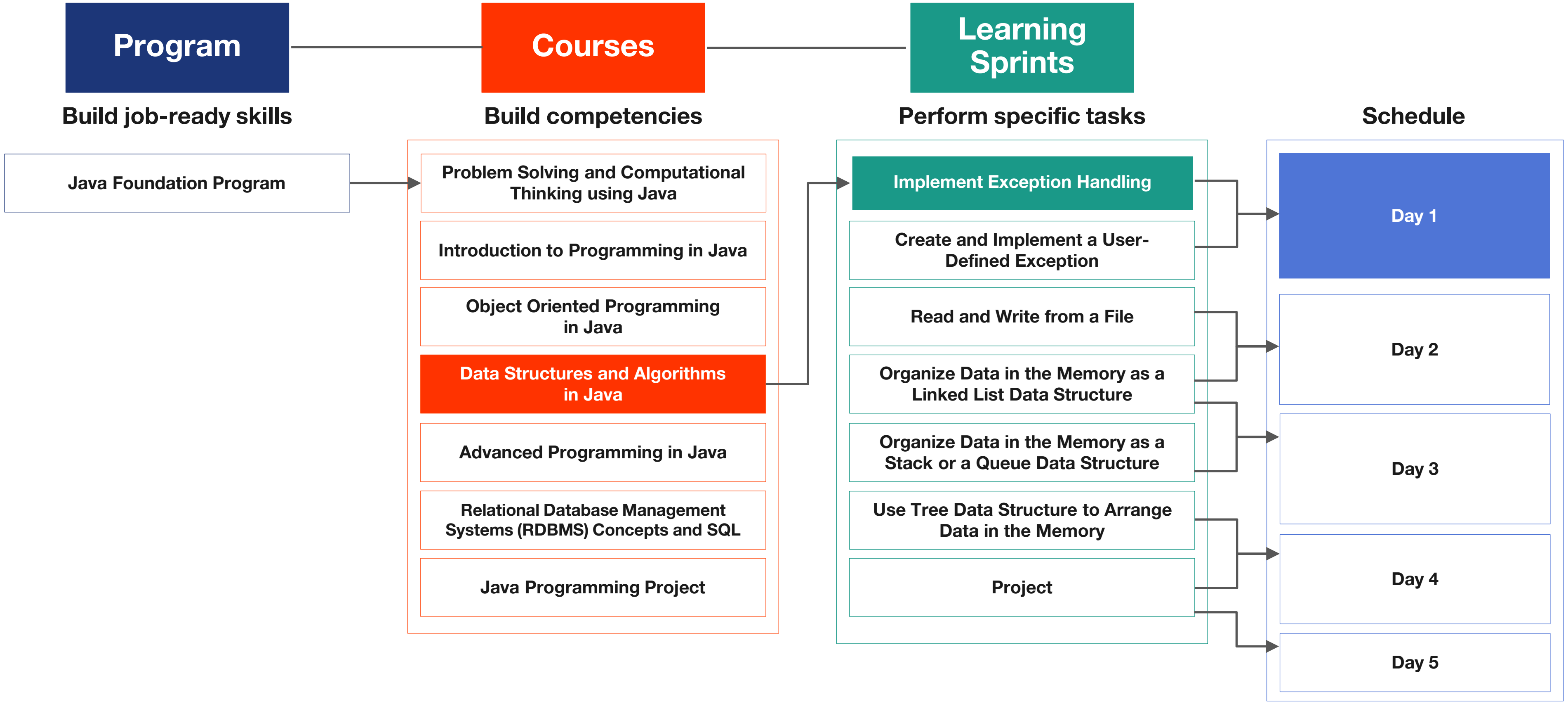
Java Program: Course 4: Plan



Program
Build job-ready skills
Java Foundation Program
Courses
Build competencies
Problem Solving and Computational Thinking using Java
Introduction to Programming in Java
Object Oriented Programming in Java
Data Structures and Algorithms in Java
Advanced Programming in Java
Relational Database Management Systems (RDBMS) Concepts and SQL
Java Programming Project
Learning Sprints
Perform specific tasks
Implement Exception Handling
Create and Implement a User-Defined Exception
Read and Write from a File
Organize Data in the Memory as a Linked List Data Structure
Organize Data in the Memory as a Stack or a Queue Data Structure
Use Tree Data Structure to Arrange Data in the Memory
Project
Schedule
Day 1
Day 2
Day 3
Day 4
Day 5

Think and Tell

There could be multiple reasons for a washing machine to stop working abruptly. Power failure, insufficient water supply, or something getting stuck in the machine could be a few.

How do you think the machine should respond in such situations?

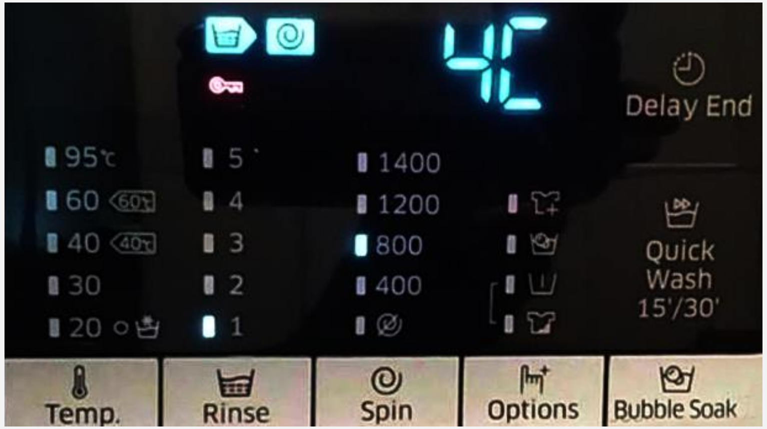
Let Us Discuss

Would you like your machine to start the wash cycle all over again or resume it from the point where it had left?

Would it help if your machine remembers the temperature, water level and other settings?

Washing Machine

Have you ever seen this code on your washing machine?



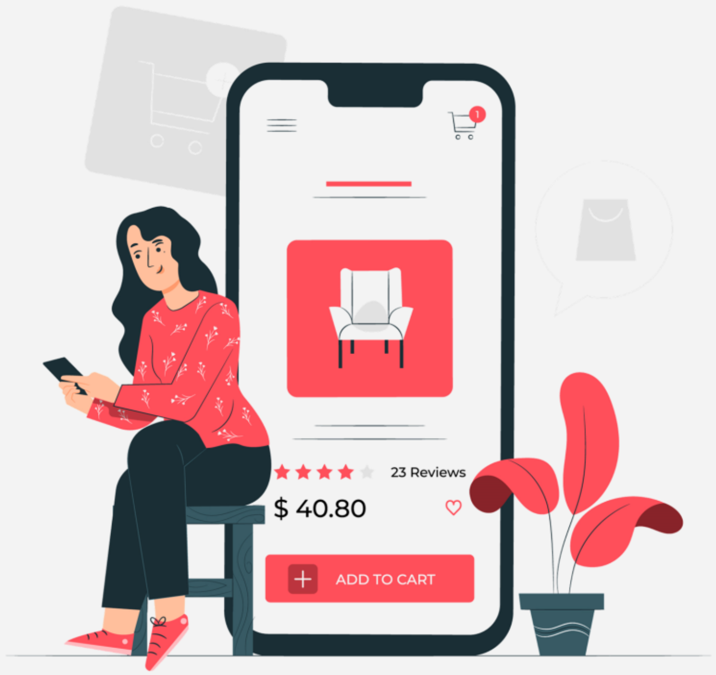
Source: https://www.republicworld.com/

Online Purchase

After spending an hour browsing the web and selecting a product to be purchased online, you try to make the payment.

Imagine how you would feel if the system tells you that the product is out of stock.

What would have been a better way for the system to respond?



Let Us Discuss

What message would you as a user wish to see is such situations?

When do you think receiving such messages would help the user the most? Give reasons.

Implementing Exception Handling

Learning Objectives

- Define errors in coding
- Describe and list the types of exceptions
- Explain the purpose and advantages of exception handling
- Present the class hierarchy of exception
- Use try, catch and finally blocks

Syntax Errors While Coding

Syntax Errors:

```
int a[] = new in[];

System.out.println("Welcome")
```

Is there any syntax error in this code?

Syntax errors occur when language rules are not followed

Errors During Coding

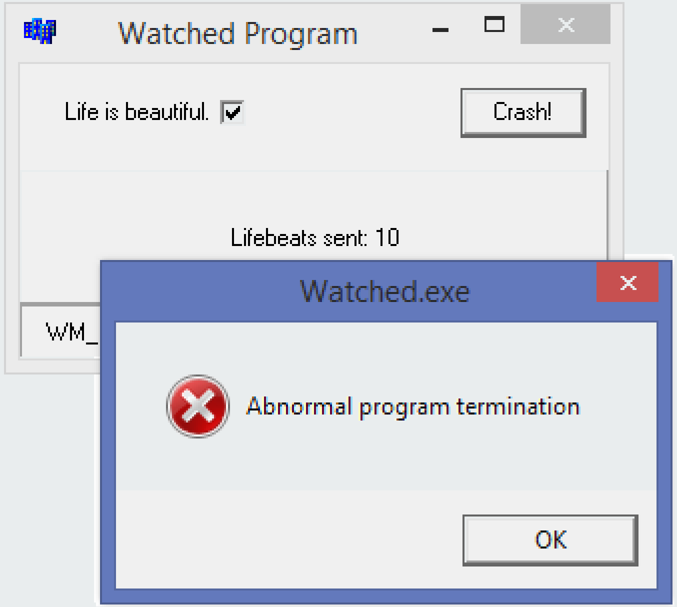
Are there any logical errors in this code?

Logical errors do not allow a program to behave as expected

Logical Errors:

```
public void
sortInAscendingOrder(int[]arr)
{
int temp;
for (int i = 0; i < arr.length; i++)
{
for(int j=i;j<arr.length;j++)
{
if(arr[i]>arr[j])
{
          temp = arr[i];
          arr[i] = arr[j];
          arr[j] = arr[i];
}
}
}
}
```

Execution Errors


Watched Program

Life is beautiful. ✓

Crash!

Lifebeats sent: 10

WM_

Watched.exe

Abnormal program termination

OK

Execution time errors are encountered during the execution of an application and causes abnormal termination of a program

Let Us Discuss

How can we rectify the following errors?

- Syntax errors
- Logical errors
- Execution time errors

Exceptions

- Exception is an object in Java. It describes an exceptional condition that occurs in a piece of code

- An exception is an unwanted event that occurs during the execution of a program and disrupts the normal flow of the program

- An exception can happen either during the Compilation or Execution phase

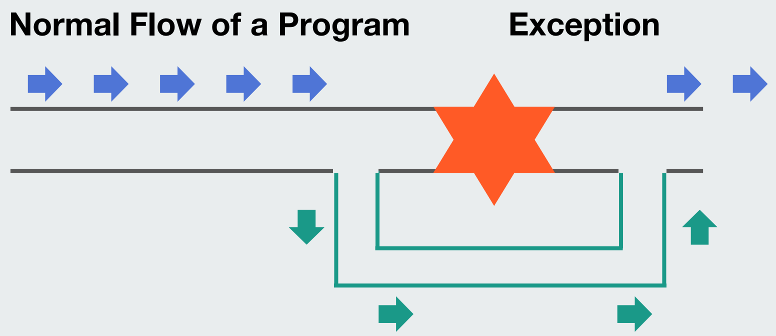
Handling Exceptions

How can we handle exceptions that occur during the compilation and execution phase?

Is there a way to handle exceptions in Java?

Exception Handling

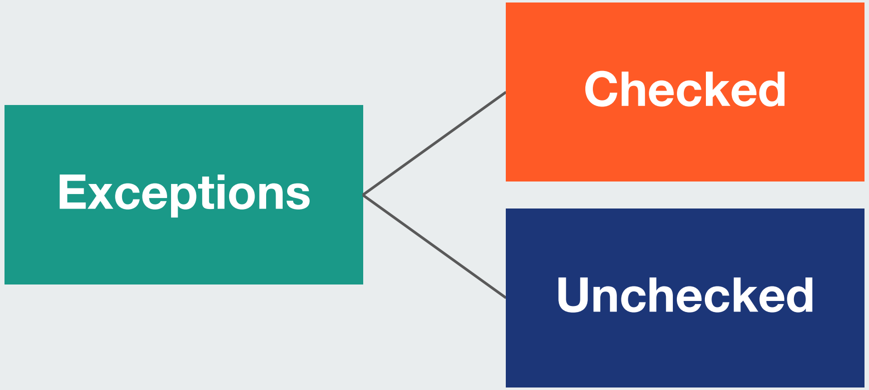
Exception handling is a mechanism that helps a programmer to bypass an exceptional situation and allows the program to flow instead of getting terminated abruptly



Advantages of Exception Handling

- A program with exception handling does not stop abruptly. It terminates gracefully after giving an appropriate message
- It helps maintain the normal flow of an application
- It separates an error handling code from a regular code
- It reports meaningful errors
- It simplifies tracing the location of errors

Types of Exceptions



There are two types of Exceptions in Java

- Checked or Compile time exception
- Unchecked or Runtime exceptions

Checked Exception

- Checked exceptions are exceptional scenarios that we can anticipate in a program and try to recover from it

- We should catch the exception and provide a meaningful message to the user and log it properly for debugging purposes

- If a checked exception is thrown
    - We should either handle it with the same method or
    - We should propagate it to the caller

Unchecked Exception

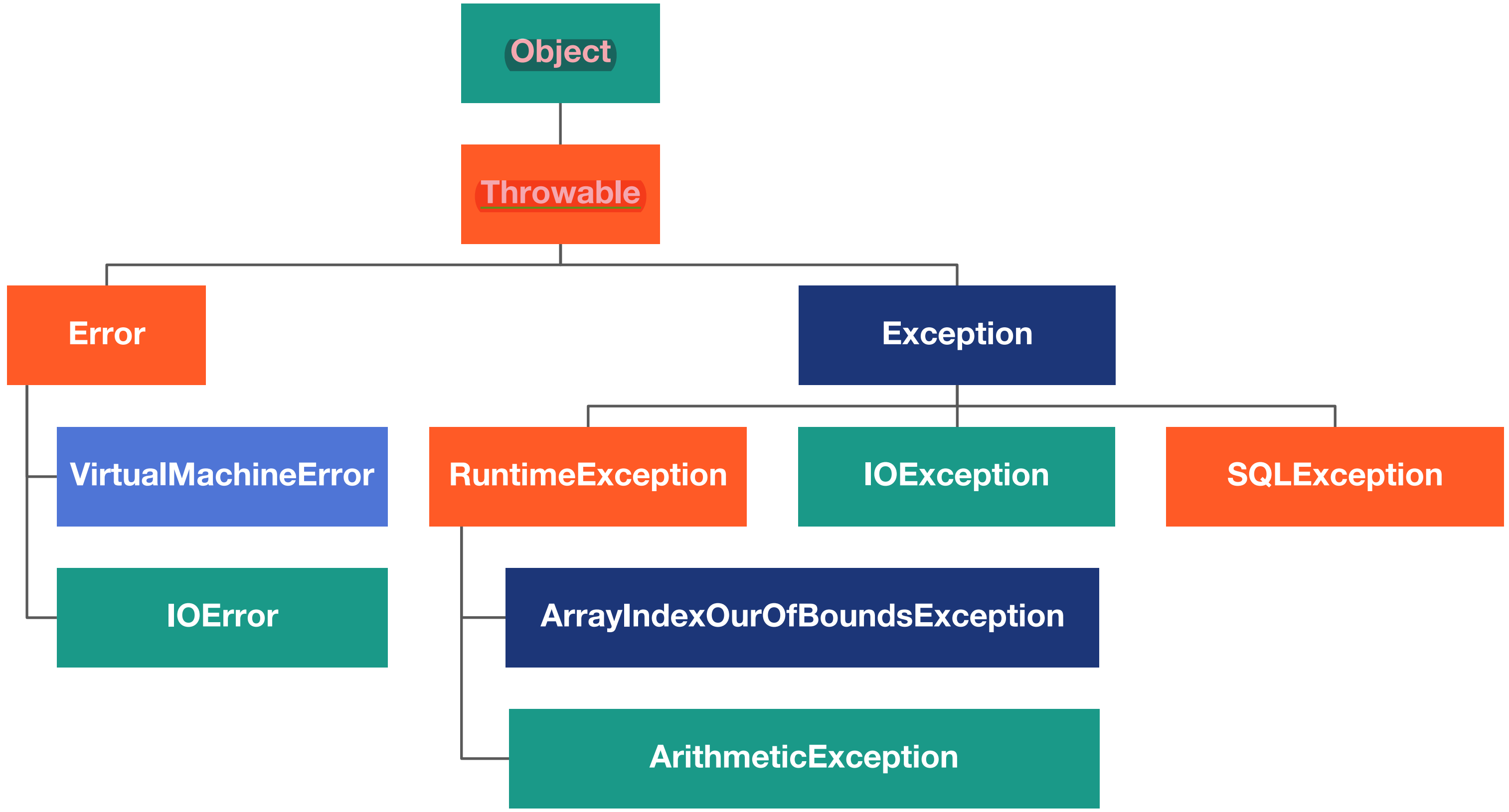
- Runtime Exceptions or Unchecked exceptions are causes of bad programming

For example,

While trying to retrieve an element from an array, the length of the array must be checked first before trying to retrieve the element, otherwise it might throw `ArrayIndexOutOfBoundException` at runtime or execution time

- Runtime Exception can be avoided with better programming

Exception Class Hierarchy



Java Code

- The Java code we write can be divided into two:
  - **Normal code** – this code does not generate any exceptional situation. Variable declarations, print statements, etc., are few examples of a normal code
  - **Critical code** – this code is likely to generate an exceptional situation. Reading a file, a mathematical operation, reading an array, etc., can produce exceptional situations

Handling Exceptions – the `try` and `catch` blocks

Handlers in Java

Following blocks are used to handle exceptions in Java:

- `try-catch`
- `try-catch-finally`

try and catch Blocks

- The try statement allows you to define a critical code that is likely to generate an exception
- The catch statement allows you to define a block of code to be executed, if an exception occurs in the try block
- The try block is followed by a catch block:

```
try {
// Critical Code
}
catch(Exception e) {
  //  Block of code to handle errors
}
```

finally Block

```
try {
// Critical Code
}
catch(Exception e) {
  //  Block of code to handle
errors
}
finally{//Block of code
}
```

Code in the finally block in Java is executed whether an exception occurs or not

A finally block follows a try or a catch block

Exception Handling

Write a program that accepts students' details and performs the following tasks:

1. Find the students whose name starts with the letter 'A'.
2. Display the details of all the students.

Handle exceptions wherever necessary.

Key Takeaway

- Errors in coding
- Types of exceptions
- Exception handling and its advantages
- Exception class hierarchy
- Keywords used to handle exceptions

Thank you!