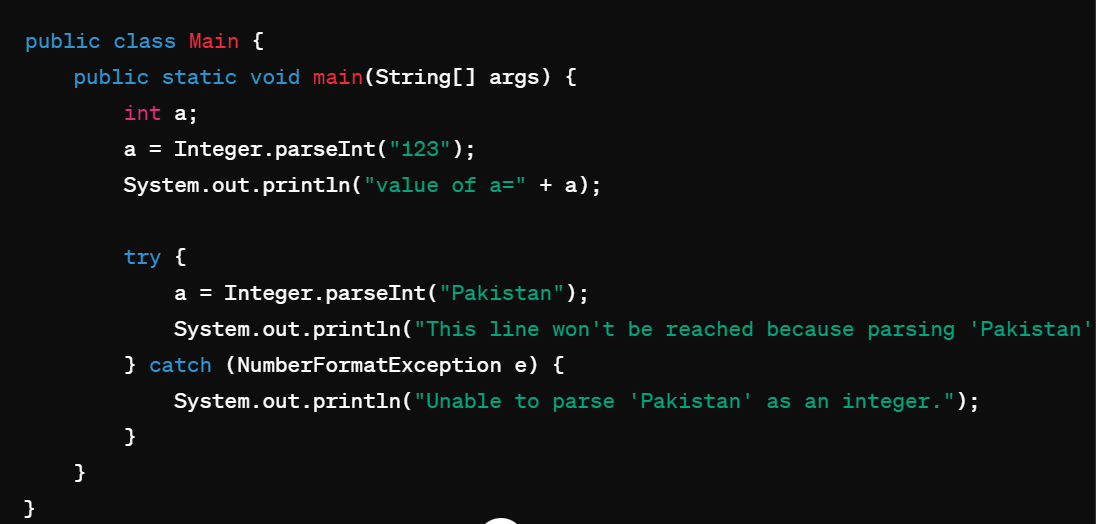
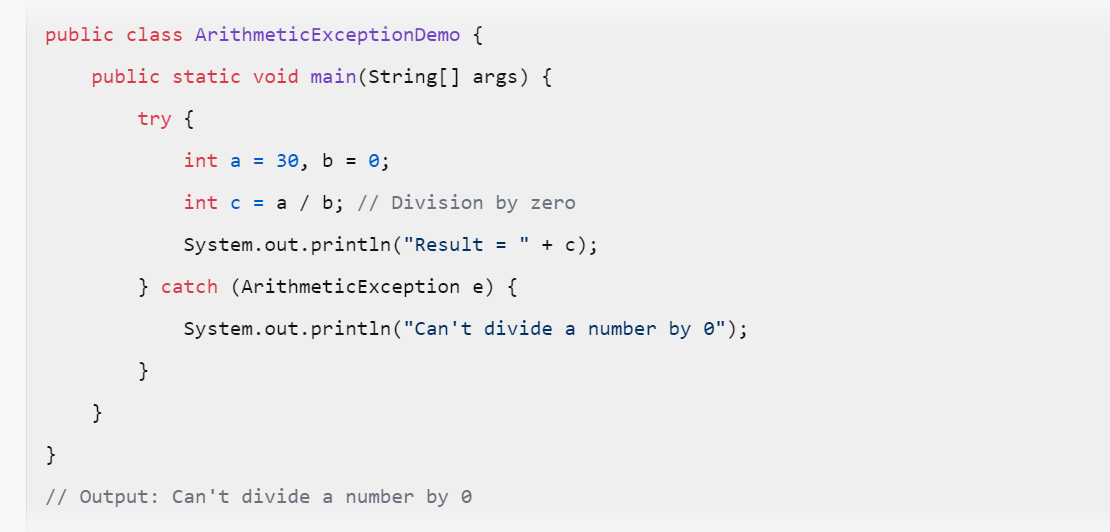
**ERRORS AND EXCEPTIONS:**

****

**TYPES OF EXCEPTIONS:**

****

****

**A screen shot of a computer code

Description automatically generated**

**A screenshot of a computer code

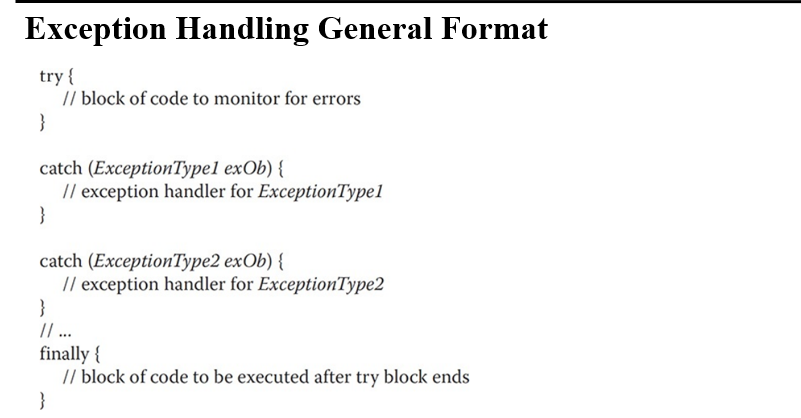
Description automatically generated**

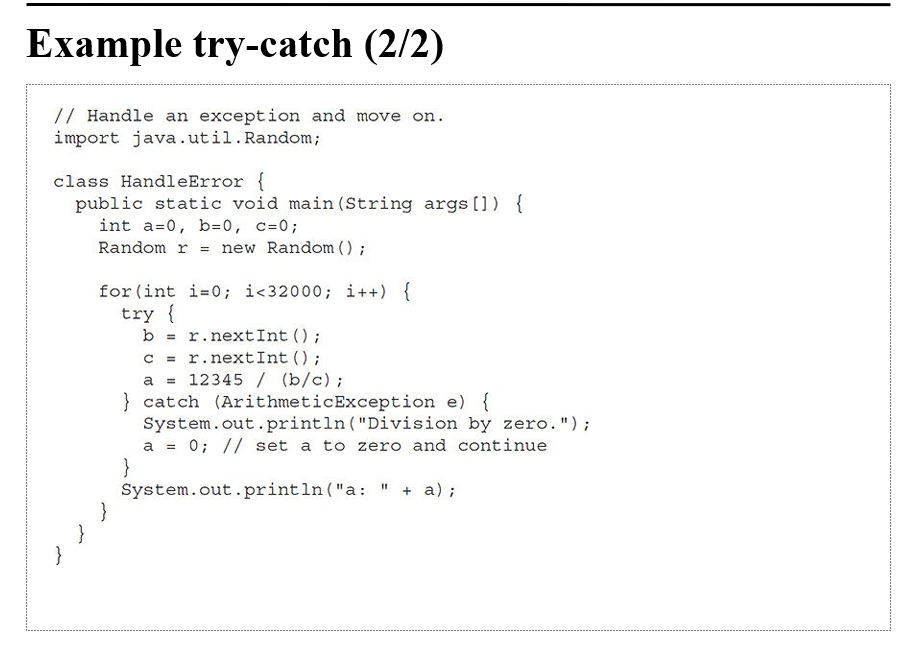
**A screen shot of a computer code

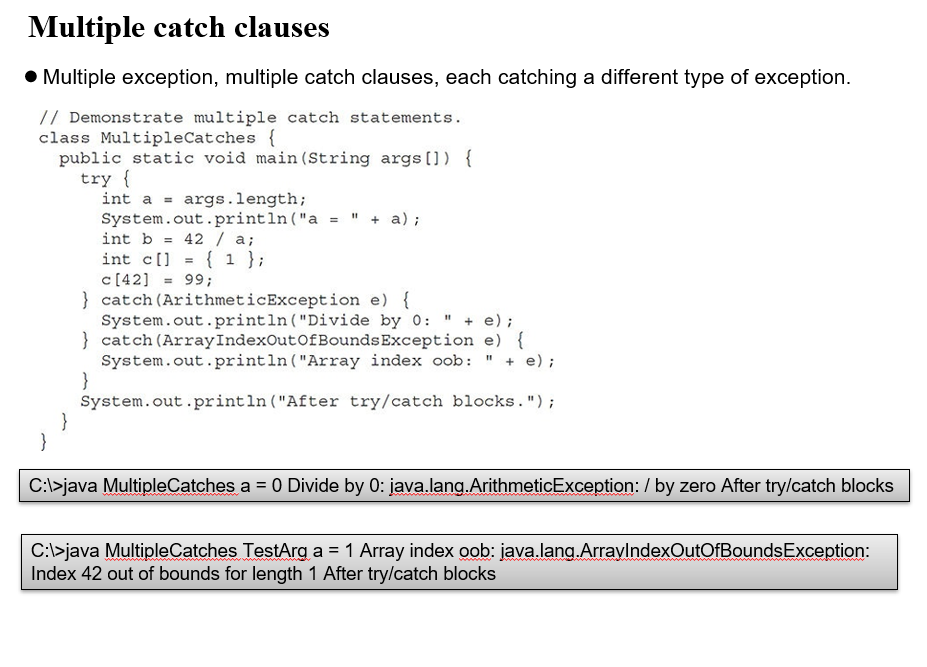
Description automatically generated**

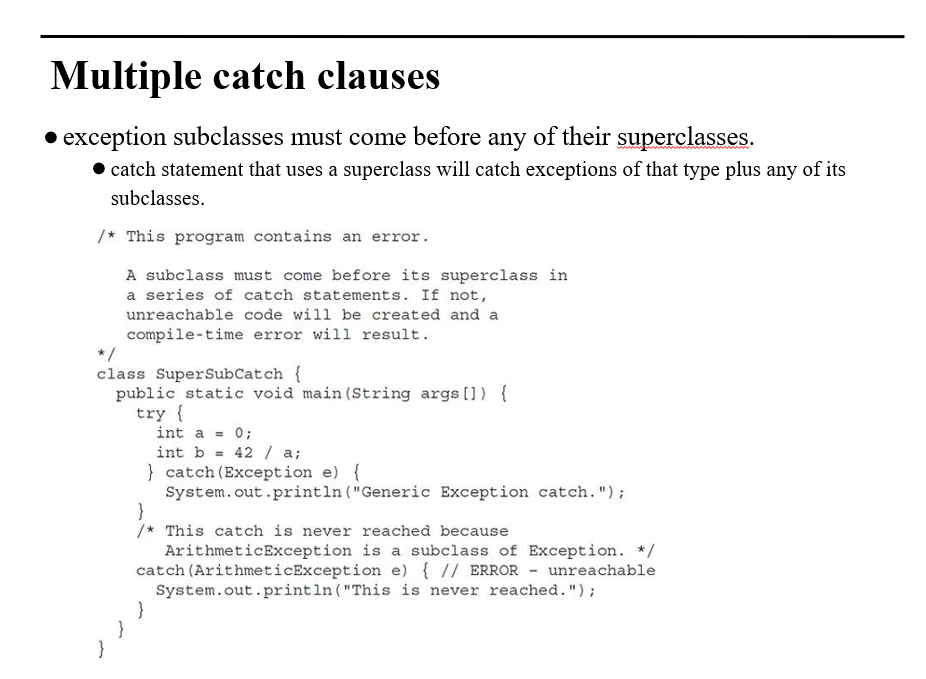
**A computer screen shot of a code

Description automatically generated**

****

****

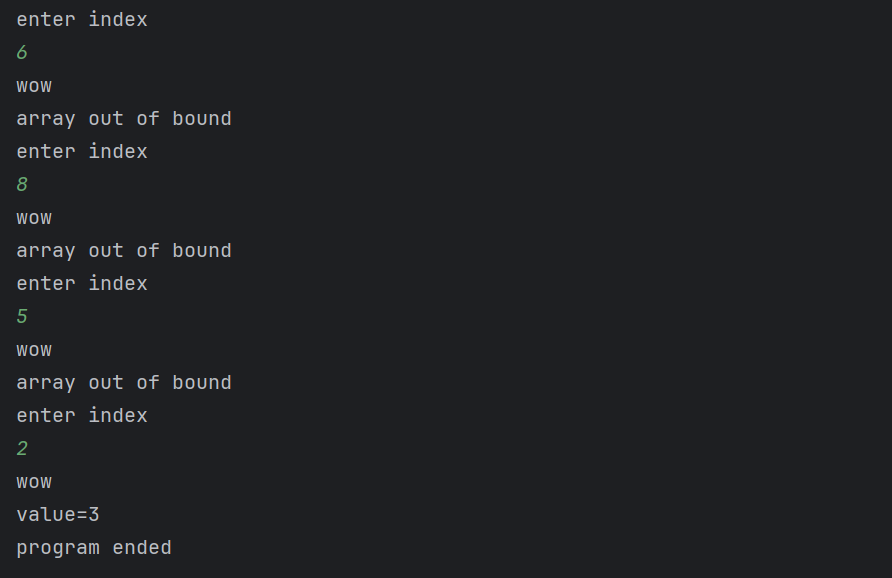
****

****

**WE ARE MAKING A PROGRAM WHICH WILL RUN UNTIL THERE IS NO EXCEPTION IN ACCESSING AN ARRAY:**

import java.util.Scanner;  
public class Main {  
 public static void main(String[] args) {  
 int[] array = new int[4];  
 //0  
 array[0] = 3;  
 array[1] = 2;  
 array[2] = 3;  
 array[3] = 4;  
 boolean flag = true;  
 while (flag) {  
 System.*out*.println("enter index");  
 Scanner s = new Scanner(System.*in*);  
 int r = s.nextInt();  
 try {  
 System.*out*.println("wow");  
 try {  
 System.*out*.println("value=" + array[r]);  
 flag=false;  
 } catch (ArrayIndexOutOfBoundsException e) {  
 System.*out*.println("array out of bound");  
 }  
 } catch (Exception n) {  
 System.*out*.println("null");  
 }  
 }  
 System.*out*.println("program ended");  
 }  
}

**OUTPUT**

****

**ASSIGNMENT:**

**Task 2:**

public class ExceptionTest {  
 int r,g,b;  
 void makeColor(int r, int g, int b) throws Exception{  
  
 if(r>255 || g>255 || b>255)  
 throw new InvalidColorException();  
 else if((r-g)>20)  
 throw new InvalidShadeException();  
 else if ((g-b)<10)  
 throw new InvalidBrightnessException();  
 }  
 String getMessage(){  
 return "Exception";  
 }  
  
 public static void main(String s[]){  
 ExceptionTest e1 = new ExceptionTest();  
 ExceptionTest e2 = new ExceptionTest();  
 ExceptionTest e3 = new ExceptionTest();  
 try {  
 e1.makeColor(260,170,165);  
 }catch (Exception e){  
 System.*out*.println("Exception : " + e.getMessage());  
 }  
 try {  
 e2.makeColor(200, 170, 165);  
 }catch (Exception e){  
 System.*out*.println("Exception : " + e.getMessage());  
 }  
 try {  
 e3.makeColor(180, 170, 165);  
 }catch (Exception e){  
 System.*out*.println("Exception : " + e.getMessage());  
 }  
 }  
  
}  
class InvalidColorException extends Exception{  
 @Override  
 public String getMessage(){  
 return "Invalid Color Exception";  
 }  
}  
class InvalidShadeException extends InvalidColorException{  
 @Override  
 public String getMessage(){  
 return "Invalid Shade Exception";  
 }  
}  
class InvalidBrightnessException extends InvalidShadeException{  
 @Override  
 public String getMessage(){  
 return "Invalid Brightness Exception";  
 }  
}

**output:**

**A black background with white text

Description automatically generated**

**OR**

public class ExceptionTest {  
 int r, g, b;  
  
 void makeColor(int r, int g, int b) throws InvalidColorException {  
 if (r > 255 || g > 255 || b > 255)  
 throw new InvalidColorException(new RuntimeException());  
 else if ((r - g) > 20)  
 throw new InvalidShadeException(new RuntimeException());  
 else if ((g - b) < 10)  
 throw new InvalidBrightnessException(“invalid brightness”);  
 }  
  
 public static void main(String[] args) {  
 ExceptionTest t = new ExceptionTest();  
  
 try {  
 t.makeColor(300, 200, 195);  
 } catch (InvalidBrightnessException e) {  
 System.*out*.println("Invalid brightness Exception: " + e.getMessage());  
 } catch (InvalidShadeException e) {  
 System.*out*.println("Invalid Shade Exception: " + e.getMessage());  
 } catch (InvalidColorException e) {  
 System.*out*.println("Invalid COLOR Exception: " + e.getMessage());  
 }  
 }  
}  
  
class InvalidColorException extends Exception {  
 InvalidColorException() {}  
  
 InvalidColorException(Throwable cause) {  
 super(cause);  
 }  
}  
  
class InvalidShadeException extends InvalidColorException {  
 InvalidShadeException(Throwable msg) {  
 super(msg);  
 }  
}  
  
class InvalidBrightnessException extends InvalidShadeException {  
 InvalidBrightnessException(Throwable msg) {  
 super(msg);  
 }  
}

**BUT msg and cause cant be together, its either msg for all or cause**

**TASK 3:**

1. **a) Can we throw an exception in the constructor of a class? why or why not ?**

Yes, you can throw an exception from the constructor of a class in Java. Constructors are not different from methods in this regard, and they can declare exceptions in their signature with the throws keyword. If a constructor throws an exception, it indicates that something went wrong during the creation of an object, and the object may not be constructed properly

1. **Validation:** Constructors can perform validation checks on input parameters or object state. If the validation fails, you might throw an exception to indicate that the object cannot be created in an invalid state.
2. **Initialization Failure:** If the constructor encounters an error during initialization that prevents the object from being properly constructed, throwing an exception can signal this failure to the caller.
3. **Resource Allocation:** Constructors that allocate resources such as file handles, network connections, or database connections might throw exceptions if the resource allocation fails

**B) Can we define some properties in a user defined exception class? If we can do it provide an example**

Yes, we can define properties in a user-defined exception class, just like in any other class in Java. This allows you to provide additional information about the exception that can be used by the catch block.

**EXAMPLE:**

public class DetailedException extends Exception {  
 private int errorCode;  
 private String errorContext;  
  
 public DetailedException(String message, int errorCode, String errorContext) {  
 super(message);  
 this.errorCode = errorCode;  
 this.errorContext = errorContext;  
 }  
  
 public int getErrorCode() {  
 return errorCode;  
 }  
  
 public String getErrorContext() {  
 return errorContext;  
 }  
}  
public class App {  
 public static void main(String[] args) {  
 try {  
 // Some operation that can fail  
 throw new DetailedException("An error occurred", 404, "Resource not found");  
 } catch (DetailedException e) {  
 System.out.println(e.getMessage());  
 System.out.println("Error Code: " + e.getErrorCode());  
 System.out.println("Error Context: " + e.getErrorContext());  
 }  
 }  
}

**OUTPUT:**

**A black background with white text

Description automatically generated**

1. **Can we make get methods in an exception class, will that be a good idea, can you identify one such scenario where a custom get message will be very helpful**

Yes, we can create getter methods in an exception class, and it can be a good idea when you want to provide more information about the exception that occurred. Getter methods allow you to retrieve the properties of an exception object, which can be particularly useful for logging, debugging, or providing detailed feedback to the user.

**EXAMPLE:**

class ValidationException extends Exception {  
 private String field;  
 private String errorMessage;  
  
 public ValidationException(String field, String errorMessage) {  
 super(errorMessage);  
 this.field = field;  
 this.errorMessage = errorMessage;  
 }  
  
 public String getField() {  
 return field;  
 }  
  
 public String getErrorMessage() {  
 return errorMessage;  
 }  
}  
  
class UserValidator {  
 public void validateUsername(String username) throws ValidationException {  
 if (username == null || username.isEmpty()) {  
 throw new ValidationException("username", "Username cannot be empty");  
 }  
 // Additional validation checks can be added here  
 }  
}  
  
public class Main {  
 public static void main(String[] args) {  
 UserValidator validator = new UserValidator();  
  
 try {  
 validator.validateUsername("");  
 } catch (ValidationException e) {  
 System.out.println("Validation failed for field: " + e.getField());  
 System.out.println("Error message: " + e.getErrorMessage());  
 }  
 }  
}

**OUTPUT:**

****

**PRACTICE:**

**Custom exception**

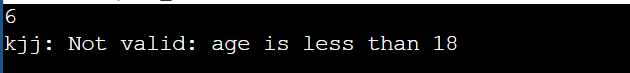
**A screen shot of a computer program

Description automatically generated**

**A black screen with white text

Description automatically generated**

**Output:**

****

* [Create a method that takes a string as input and throws an exception if the string does not contain vowels1](https://www.w3resource.com/java-exercises/exception/index.php).

public class exceptionHandling {  
 public static void main(String[] args) {  
 String s;  
 try {  
 s = "lkjh";  
 k.*string*(s); //using k.string as string is a method of k class  
 System.*out*.println("ans= " + s);  
 } catch (mkn j) {  
 System.*out*.println("bhxxvg<>" + j.getMessage());  
 }  
  
 }  
   
 //making it static as it is (must be) in class exceptionHandling   
 static class k {  
 public static void string(String l) throws mkn{  
  
 boolean b=false;  
 for( int i=0;i<l.length();i++){  
 char ch=k.charAt(i);  
 //for single char use '' not ""  
 if(ch =='a'|| ch=='e' || ch=='i' || ch=='o' || ch=='u')  
 {  
 b=true;  
 break;  
 }}  
  
//dont use if(b==false)  
 if(!b){  
 throw new mkn("kmkm");  
 }  
 }  
 }  
//making it static as it is in class exceptionHandling  
 static class mkn extends Exception{  
 mkn(String m){  
 super(m);  
 }}  
}