Java Exception Handling: Concepts and Practice

Concept: Basic try-catch block usage in exception handling.

Question 1: Write a Java program that uses a try-catch block to handle a potential ArithmeticException.

Solution 1:

```
public class Main {
    public static void main(String[] args) {
        try {
            int result = 10 / 0;
        } catch (ArithmeticException e) {
            System.out.println("Caught ArithmeticException: " +
e.getMessage());
        }
    }
}
```

Concept: Exception handling with method-specific throws declaration.

Question 2: Create a Java method that throws an IOException and use a try-catch block to handle this exception.

Solution 2:

```
import java.io.IOException;

public class Main {
    public static void main(String[] args) {
        try {
            riskyMethod();
        } catch (IOException e) {
            System.out.println("Caught IOException: " +
e.getMessage());
        }
    }

    public static void riskyMethod() throws IOException {
        throw new IOException("Simulated error");
    }
}
```

Concept: Use of the finally block in exception handling.

Question 3: Demonstrate the use of the finally block in Java to execute code regardless of whether an exception occurs.

Solution 3:

```
public class Main {
    public static void main(String[] args) {
        try {
            int result = 10 /0;
        } catch (ArithmeticException e) {
            System.out.println("Caught ArithmeticException");
        } finally {
            System.out.println("This block always executes");
        }
    }
}
```

Concept: Custom exception class creation and usage.

Question 4: Write a Java program that creates a custom exception class and throws it from a method.

Solution 4:

```
class MyCustomException extends Exception {
    public MyCustomException(String message) {
        super(message);
    }
}

public class Main {
    public static void main(String[] args) {
        try {
            throwCustomException();
        } catch (MyCustomException e) {
            System.out.println("Caught MyCustomException: " +
e.getMessage());
        }
    }

    public static void throwCustomException() throws
MyCustomException {
        throw new MyCustomException("Custom error occurred");
    }
}
```

Concept: Exception propagation through method call stack

Question 5: Create a Java program that illustrates exception propagation in methods.

Solution 5:

```
public class Main {
    public static void main(String[] args) {
        try {
            method1();
        } catch (Exception e) {
            System.out.println("Exception caught in main method: "
+ e.getMessage());
        }
    }

    public static void method1() throws Exception {
        method2();
    }

    public static void method2() throws Exception {
        throw new Exception("Exception from method2");
    }
}
```