1. الاستثناءات

الإستثناء (Exceptions) في البرمجة عبارة عن حدث يقوم بتعطيل السير الطبيعي للبرنامج. وهو كائن يتم إلقاؤه في وقت التشغيل. يمكن أن يحدث استثناء لأسباب عديدة ومختلفة منها الخطأ النحوي والخطأ وقت التشغيل والخطأ في المنطق.

1.1خطأ لغوي – Syntax Error

الخطأ اللغوي (Syntax Error) هو الخطأ في كتابة الأوامر البرمجية سواء في الكتابة بالأحرف الكبيرة أو نسيان الأقواس أو عدم استدعاء class وغيرها.

```
system.out.print;
```

1.2خطأ وقت التشغيل - Runtime error

يحدث الخطأ في وقت التشغيل عندما يكون البرنامج صحيحًا من الناحية اللغوية، ولكنه يحتوي على مشكلة يتم اكتشافها فقط أثناء تنفيذ البرنامج.

```
class DivByZero {
    public static void main(String args[])
    {
        int var1 = 15;
        int var2 = 5;
        int var3 = 0;
        int ans1 = var1 / var2;

        // This statement causes a runtime error,
        // as 15 is getting divided by 0 here
        int ans2 = var1 / var3;

        System.out.println(
                "Division of val1"
                        + " by var2 is: "
                        + ans1);
        System.out.println(
                "Division of val1"
                        + " by var3 is: "
                        + ans2);
    }
}
```

1.3خطأ منطقي - Logical error

الخطأ المنطقي (Logical Error) هي الأخطاء التي يرتكبها المبرمجون. تعمل هذه البرامج التي بها هذه الأخطاء ولكنها لا تعطي النتائج المتوقعة.

```
class IncorrectMessage {
    public static void main(String args[])
    {
        System.out.println(sum(6,3));
    }
    public static int sum(int num1,int num2){
        return num2-num1;
    }
}
```

1.4 حل استثناءات وقت التشغيل - Solving Runtime Exceptions

try-catch 1.4.1

يتم استخدام try-catch في جافا لإحاطة التعليمات البرمجية التي قد تؤدي إلى إستثناء. يجب استخدامه ضمن الميثود. إذا حدث استثناء في جملة معينة في try block، فلن يتم تنفيذ باقي الكود. لذلك، يوصى بعدم الاحتفاظ بالكود في try-block التي لن تؤدي إلى استثناء. يجب أن يتبع Java try block إما catch أو block.

```
try {
        int[] myNumbers = {1, 2, 3};
        System.out.println(myNumbers[10]);
        } catch (Exception e) {
        System.out.println("Something went wrong.");
        }
```

throw & throws 1.4.2

تعريف throws

إذا قمت بتعريف دالة وأردت لهذه الدالة أن ترمي إستثناء إذا حدث شيء معين فعليك وضع الكلمة throws بعد أقواس الباراميترات ثم تحديد نوع الإستثناء الذي قد ترميه الدالة، وإذا قمت مسبقاً بتعريف إستثناء يمكنك جعل الدالة تقوم برميه.

```
public class Main {
    static void checkAge(int age) throws ArithmeticException {
        if (age < 18) {
            throw new ArithmeticException("Access denied - You must
be at least 18 years old.");
        }
        else {
            System.out.println("Access granted - You are old
enough!");
        }
    }

    public static void main(String[] args) {
        checkAge(15); // Set age to 15 (which is below 18...)
    }
}
```

تعريف throw

يتم استخدام الكلمة throw لإنشاء خطأ مخصص، يتم استخدام تعليمة throw مع نوع الاستثناء.

```
public class Main {
    static void checkAge(int age) {
        if (age < 18) {
            throw new ArithmeticException("Access denied - You must
be at least 18 years old.");
        }
        else {
            System.out.println("Access granted - You are old
enough!");
        }
    }

    public static void main(String[] args) {
        checkAge(15); // Set age to 15 (which is below 18...)
    }
}
```

1.5 أنواع استثناءات وقت التشغيل - Types of Runtime Exceptions

Checked Exceptions 1.5.1

هذه هي الاستثناءات التي يتم التحقق منها في وقت الترجمة. إذا ألقى بعض الكود ضمن طريقة استثناءً محددًا، فيجب إما معالجة الاستثناء أو تحديد الاستثناء باستخدام الكلمة الأساسية throws.

```
// Java Program to Illustrate Checked Exceptions
// Where FileNotFoundException does not occur
import java.io.*;

class GFG {

    // Main driver method
    public static void main(String[] args)
            throws IOException
    {
        // Creating a file and reading from local repository
        FileReader file = new FileReader("C:\\test\\a.txt");

        // Reading content inside a file
        BufferedReader fileInput = new BufferedReader(file);

        // Printing first 3 lines of file "C:\test\a.txt"
        for (int counter = 0; counter < 3; counter++)
            System.out.println(fileInput.readLine());

        // Closing all file connections
        // using close() method
        // Good practice to avoid any memory leakage
        fileInput.close();
    }
}
```

Unchecked Exceptions 1.5.2

هذه هي الاستثناءات التي لم يتم التحقق منها في وقت compile. في استثناءات Java ضمن فئات Error وRuntimeException هي استثناءات لم يتم تحديدها، يتم التحقق من كل شيء آخر ضمن قابل للإلقاء.

```
// Java Program to Illustrate Un-checked Exceptions

// Main class
class GFG {

    // Main driver method
    public static void main(String args[])
    {
        // Here we are dividing by 0
        // which will not be caught at compile time
        // as there is no mistake but caught at runtime
        // because it is mathematically incorrect
        int x = 0;
        int y = 10;
        int z = y / x;
    }
}
```