2. What are checked and unchecked exceptions? When do you use them? Explain those using examples in code(2 marks).

ANSWER:

Checked Exceptions are those exceptions that are checked at compile time. An exception is a checked exception if it inherits from java.lang.Throwable

Example:

This code won't compile because it can throw a checked exception

```
public void ioOperation(boolean isResourceAvailable) {
    if (!isResourceAvailable) {
        throw new IOException();
    }
}
```

To sort this type of error, we can try catching the exception and handling it, or by declaring that the exception can be thrown using the throws keyword.

An example of catching the errors:

```
public void ioOperation(boolean isResourceAvailable) {
    try {
        if (!isResourceAvailable) {
        throw new IOException();
        }
    } catch(IOException e) {
        // Handle caught exceptions.
    }
}
```

An example declaring the exception

```
public void ioOperation(boolean isResourceAvailable) throws IOException {
    if (!isResourceAvailable) {
        throw new IOException();
    }
}
```

Unchecked are the exceptions that are not checked at compile time. This requires the programmer to specify or catch the exceptions.In Java exceptions under java.lang.Error and java.lang.RuntimeException classes are unchecked exceptions.

These are exceptions that can be thrown without being caught or declared They are usually related to hard-coded issues like data errors, arithmetic overflow, divide by zero etc.

This Java program compiles fine, but it throws ArithmeticException when run. The compiler allows it to compile,
because ArithmeticException is an **unchecked exception.**

```
class Main {
    public static void main(String args[]) {
        int x = 0;
        int y = 10;
        int z = y/x;
    }
}
```