Exercises

Try to resolve manually, then program the code to discover the answers

Exercício: qual seria a saída do programa abaixo? E se a terceira linha do programa fosse alterada para String test = 'yes';?"

```
public class TestExceptions {

  public static void main(String[] args) {
    String test = "no";
    try {
      System.out.println("start try");
      doRisky(test);
      System.out.println("end try");
    } catch (ScaryException se) {
      System.out.println("scary exception");
    } finally {
      System.out.println("finally");
    }
    System.out.println("end of main");
  }

  static void doRisky(String test) throws ScaryException {
    System.out.println("start risky");
    if ("yes".equals(test)) {
      throw new ScaryException();
    }
    System.out.println("end risky");
  }
}

class ScaryException extends Exception {
}
```

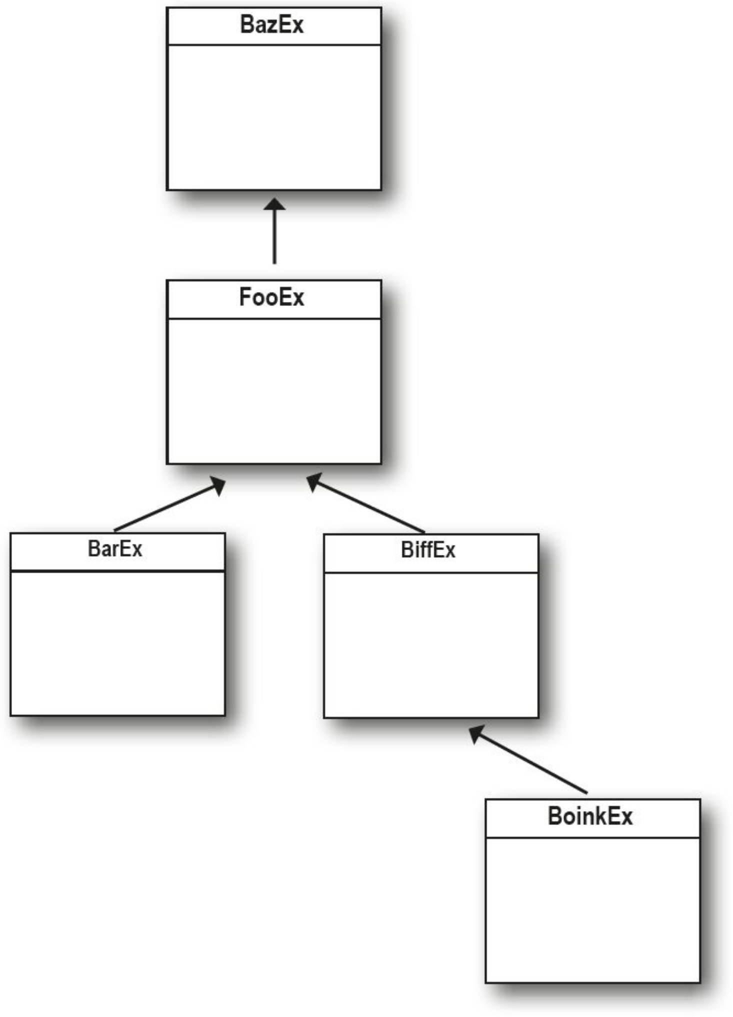
Output when `test = "no"`

Output when `test = "yes"`

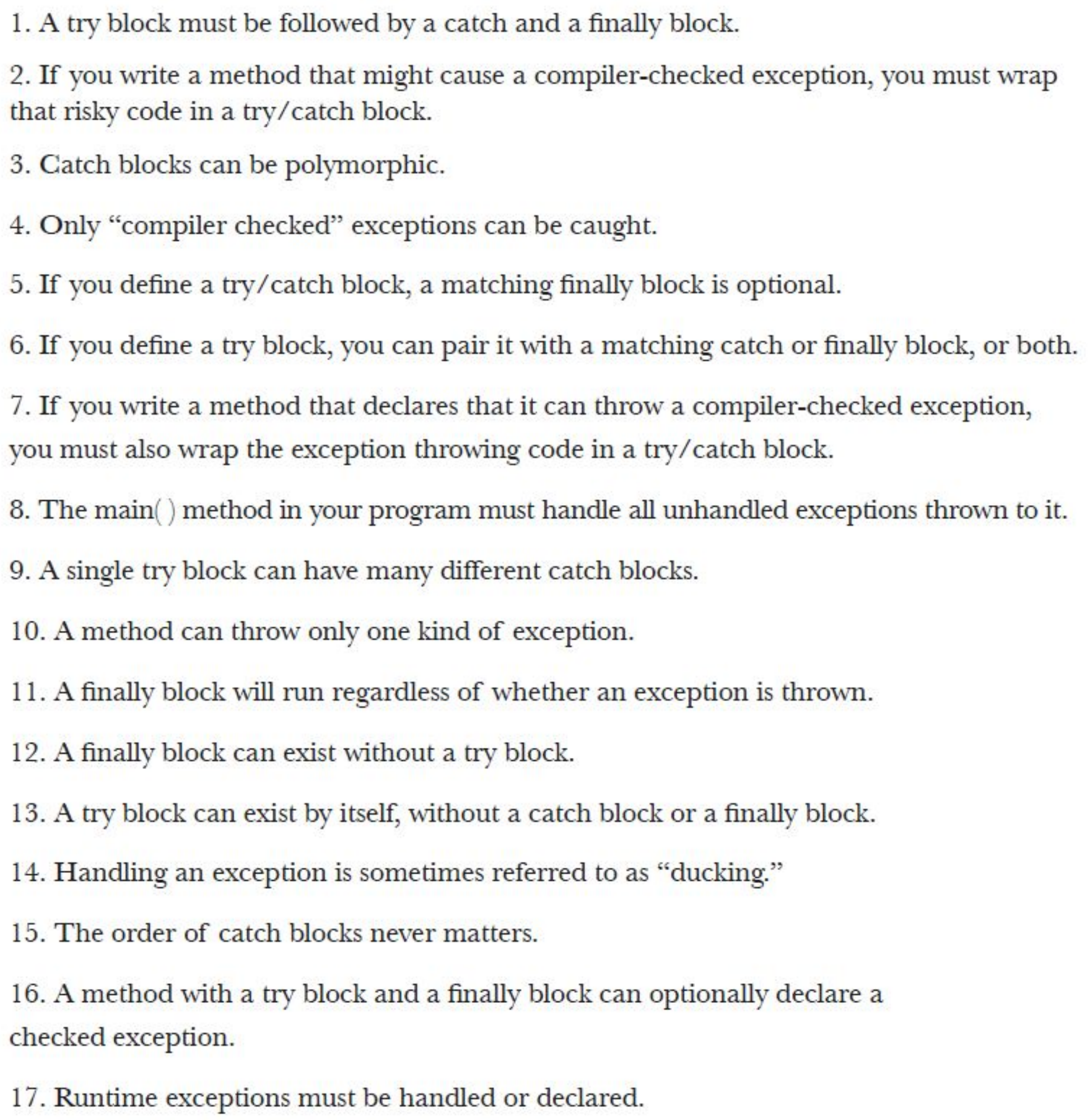
Exercício: quais estruturas de herança de classes tornariam os blocos try/catch no código abaixo correto? Desenhe um diagrama com a hierarquia correta.

```
try {
  x.doRisky();
} catch(AlphaEx a) {
  // recovery from AlphaEx
} catch(BetaEx b) {
  // recovery from BetaEx
} catch(GammaEx c) {
  // recovery from GammaEx
} catch(DeltaEx d) {
  // recovery from DeltaEx
}
```

Exercício: escreva duas estruturas try/catch diferentes que representem com precisão o diagrama abaixo. Assuma que TODAS essas exceções podem ser lançadas pelo método com o bloco try/catch.



Exercício: verdadeiro ou falso?



1. A try block must be followed by a catch and a finally block.

2. If you write a method that might cause a compiler-checked exception, you must wrap that risky code in a try/catch block.

3. Catch blocks can be polymorphic.

4. Only "compiler checked" exceptions can be caught.

5. If you define a try/catch block, a matching finally block is optional.

6. If you define a try block, you can pair it with a matching catch or finally block, or both.

7. If you write a method that declares that it can throw a compiler-checked exception, you must also wrap the exception throwing code in a try/catch block.

8. The main( ) method in your program must handle all unhandled exceptions thrown to it.

9. A single try block can have many different catch blocks.

10. A method can throw only one kind of exception.

11. A finally block will run regardless of whether an exception is thrown.

12. A finally block can exist without a try block.

13. A try block can exist by itself, without a catch block or a finally block.

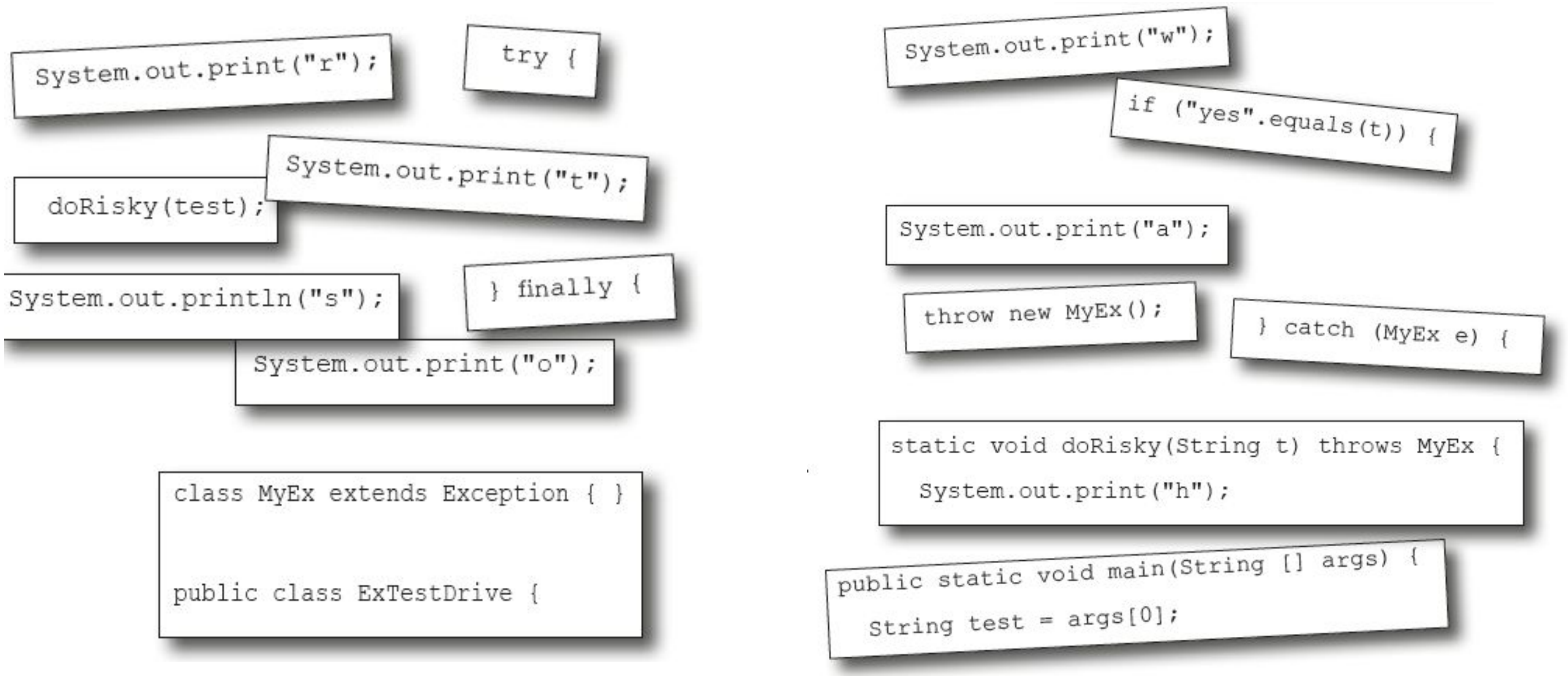
14. Handling an exception is sometimes referred to as "ducking."

15. The order of catch blocks never matters.

16. A method with a try block and a finally block can optionally declare a checked exception.

17. Runtime exceptions must be handled or declared.

Exercício: use os trechos de código para criar um programa Java funcional que produza a saída listada abaixo (você precisa adicionar as chaves faltantes).



```
System.out.print("r");
```

```
try {
```

```
System.out.print("t");
```

```
doRisky(test);
```

```
System.out.println("s");
```

```
} finally {
```

```
System.out.print("o");
```

```
class MyEx extends Exception { }


public class ExTestDrive {
```

```
System.out.print("w");
```

```
if ("yes".equals(t)) {
```

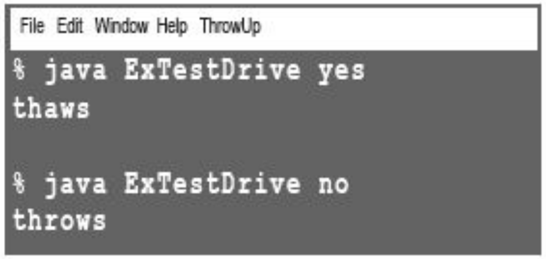
```
System.out.print("a");
```

```
throw new MyEx();
```

```
} catch (MyEx e) {
```

```
static void doRisky(String t) throws MyEx {
  System.out.print("h");
```

```
public static void main(String [] args) {
  String test = args[0];
```



```
% java ExTestDrive yes
thaws

% java ExTestDrive no
throws
```