Операторы и структура кода. Исключения

Рассматриваемые вопросы

1. Управление ходом выполнения программы
2. Блоки и инструкции
3. Исключения Java

Управление ходом программы

Выполнение оператора может быть прервано, если в потоке вычислений будут обнаружены операторы

- break
- continue
- return
- throw

Нормальное и преждевременное завершение инструкций

Единственная причина, по которой выражение может быть завершено преждевременно — генерация исключения, или из-за инструкции **throw** с заданным значением, или из-за исключения или ошибки времени выполнения.

Локальные классы и переменные

Локальным классом является вложенный класс, не являющийся членом никакого класса и имеющий имя.

Инструкция объявления **локальной переменной** объявляет одно или несколько имен локальных переменных.

Условные операторы

```
if (логическое выражение)
        //блок, логическое выражение истинно
else
        //блок, если логическое выражение ложно
```

Условные операторы

```
switch (выражение) {
    case значение1:
        //последовательность операторов
        break;
    case значение2:
        //последовательность операторов
        break;
    ...
    default:
        //последовательность операторов
}
```

Операторы цикла

```
while (условие) {

    //тело цикла

}
```

Операторы цикла

```
do {
    //тело цикла
} while ( условие );
```

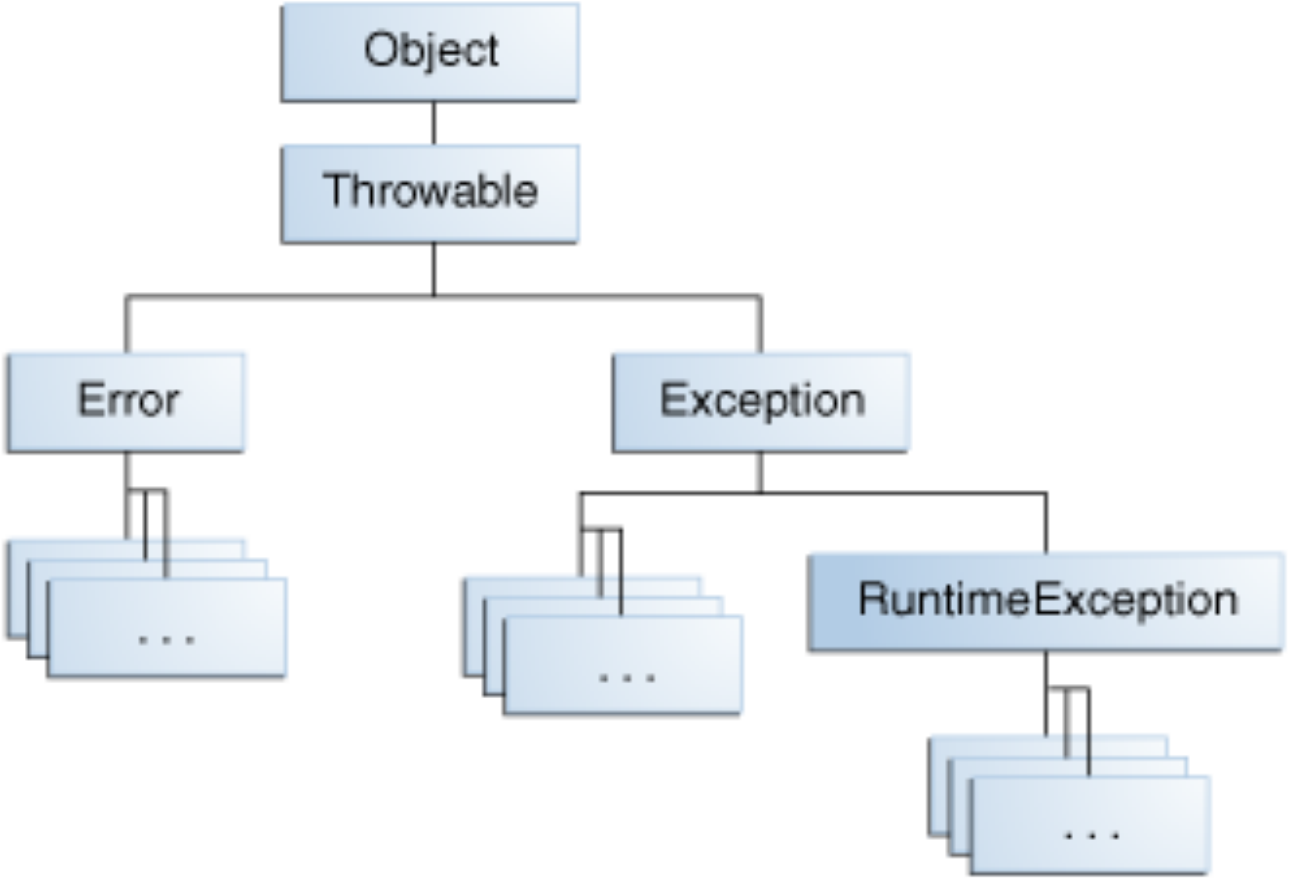
Операторы цикла

```
for (инициализация; условие; итерация) {
    //тело цикла
}
```

Цикл в стиле foreach:

```
for( тип переменная: коллекция ) {
    //тело цикла
}
```

Исключения



Исключения

Error – критическая ошибка, приведшая к остановке программы.

Exception - общий класс исключений.

Исключения бывают «проверяемыми» и «непроверяемыми».

Исключения

Проверяемые исключения:

- Унаследованные непосредственно от класса Exception (кроме RuntimeException)
- Унаследованные далее по иерархии наследования от Exception (кроме RuntimeException)

Исключения

```
try {
    ...
} catch(SomeExceptionClass e) {
    ...
} catch(AnotherExceptionClass e) {
    ...
} finally {
 ...
}
```

Исключения

```
static String readFirstLineFromFile(String path) throws IOException {


    try ( BufferedReader br = new BufferedReader(
            new FileReader(path)) ) {
        return br.readLine();
    }


}
```

Исключения

Переопределяемый метод объявляет список возможных исключений, то переопределяющий метод не может расширять этот список, но может его сужать.

Исключения

```
public class BaseClass {
    public void method () throws Exception {
   ...
    }
}

public class LegalOne extends BaseClass {
    public void method () throws IOException {
      ...
    }
}
```

Вопросы для самоконтроля

проанализируйте результат выполнения программы

```
class A{
    public A method() throws Throwable { // 1
        return new Single();
    }
}

class Single extends A{
    public Single method(String str) throws RuntimeException { // 2
        return new Single();
    }
    public Single method() throws IOException {  //3
        return new Double();
    }
}

class Double extends Single{
    public void method(Integer digit) throws ClassCastException {      // 4
    }
    public Double method() throws Exception {  // 5
        return new Double();
    }
}
```

Вопросы для самоконтроля

проанализируйте результат выполнения программы

```
public class Main {
    public static void main(String[] args) {
        try{
            String str = null;
            if(str.equals("message")){
                System.out.println(str);
            }
        } catch (NullPointerException npe){
            System.out.println("NPE");
            return;
        } catch (ArithmeticException are){
            System.out.println("ARE");
        } catch (Exception ex){
            System.out.println("EX");
        } finally {
            System.out.println("Finally");
        }
    }
}
```