**Тест №4. Исключения и сборщик мусора**

1. Оператор, используемый для самостоятельного возбуждения исключительных ситуаций в Java:

**- throw**

- catch

- runtime

- try Исключения. Самостоятельное возбуждение исключительных ситуаций.

1. Обработка исключения в Java производится с помощью операторов:

- throw

**- try…catch**

- throws

- runtime Исключения. Обработка исключений.

1. Применяя данный метод можно определять специальные действия, которые будут выполняться тогда, когда объект будет использоваться сборщиком мусора:

**- finalize()**

- object()

- throwable()

- нет верного ответа. Сборка мусора. Finalize в Java.

1. Выберите верное утверждение относительно throw:

- после слова throw записывается объект класса-исключения;

- throw выбрасывает записанный в нем объект-исключение и дальше обработка этого исключения идет как обычно;

-используется для самостоятельного возбуждения исключительных ситуаций;

**- все варианты.** Исключения. Использование оператора throw.

1. Дана программа на Java:

class ChisloException extends Exception{

private String msg;

ChisloException(){ msg = null;}

ChisloException(String s){ msg = s;}

public String toString(){

return "ChisloException (" + msg + ")";

}

}

class ExceptDemo {

static public void handle(intсhislo) throws ChisloException{

System.out.println("Начало handle() для "+сhislo);

if (сhislo< 0 || сhislo> 9)

throw new ChisloException("" + сhislo);

System.out.println("Конец handle() для "+сhislo);

}

public static void main(String[] args){

try{

handle(1) ;

handle(10);

}

catch(ChisloExceptionce){

System.out.println("Исключение " + ce) ;

ce.printStackTrace();

}

}

}

В результате получим:



Что будет выведено вместо знака «?»

- 0; 9; 10.

**- 1; 10; ChisloException (10).**

- 10; 1; ChisloException (1).

- 1; 9; ChisloException (10). Исключения. Выполнение программы.

1. Сборка мусора (garbage collection) – это…

**-процесс освобождения памяти посредством удаления объектов, которые уже не будут востребованы приложениями;**

- динамическая среда, поддерживающая исполнение программы;

- ручное удаление объектов, на которые нет ссылок из программы;

- ни один из вариантов. Механизм сборки мусора. Основные принципы.

1. Выберите верные утверждения относительно блока finally:

**- finally{} выполняется независимо от того, произошло исключение или нет;**

**- если в операторе обработки исключений есть блок finally{}, то блок catch(){} может отсутствовать;**

- все, что написано в блоке finally{}, не будет выполнено, если выход из блока try{} осуществляется оператором return;

- все варианты ответов. Обработка исключительных ситуаций Конструкция try-catch-finally.

1. Ключевое слово throws указывает, что:

-**метод потенциально может создать/вызвать исключительную ситуацию;**

- метод потенциально может обработать исключительную ситуацию;

- **метод потенциально может выбросить исключение с указанным типом;**

- все варианты ответов. Исключения. Использование оператора throws.

1. В каком случае объект в Java становиться приемлемым для сборщика мусора?

- Если на данный объект есть ссылки.

**- Если на данный объект нет никаких ссылок.**

**- Если объекту было назначено нулевое значение.**

- Нет верного ответа. Сборка мусора. Механизм работы.

1. Сколько объектов будут подлежать сборке мусора после строки 7?

public class TutorialGC  
  {  
    public static void main(String [] args)  
    {  
      Object a = new Integer(100);  // Line1  
      Object b = new Long(100);     // Line2  
      Object c = new String("100"); // Line3  
      a = null;                     // Line4  
      a = c;                        // Line5  
      c = b;                        // Line6  
      b = a;                        // Line7  
      // Rest of the code here  
    }  
  }

- 0.

**- 1.**

- 2.

- 3.

- Код не возможно скомпилировать. Механизм сборки мусора. Выполнение программы.