Практическая работа №8

Теория

В **Java API** существует множество предопределенных классов исключений. **Исключения** – это объекты, а объекты определяются с помощью классов. Корневым классом для исключений является `java.lang.Throwable`. Классы исключений можно разделить на три основных типа: системные ошибки, исключения и ошибки во время выполнения. В **Java** можно **определить пользовательский класс исключений**, породив его от класса `java.lang.Exception`.

Описание Throwabe
Описание Exception
Спецификация Exception в языке Java

Системные ошибки

Системные ошибки представлены в классе Error, их выбрасывает JVM. **Класс Error** описывает внутренние ошибки системы, хотя такие ошибки и происходят очень редко. Если все-таки они происходят, то мало что можно сделать, кроме уведомления пользователя и попытки завершить программу.

Класс	Возникновение исключения
LinkageError	У одного класса есть некое отношение с другим классом, но последний класс несовместимо изменился после компиляции первого класса.
VirtualMachineError	JVM неисправна или исчерпала ресурсы, необходимые для продолжения работы.

Исключения

Исключения представлены в классе **Exception**, который описывает ошибки, вызванные самой программой и внешними обстоятельствами. Эти ошибки можно обнаружить и обработать с помощью программы.

Класс	Возникновение исключения
	Попытка использовать несуществующий класс. Это

ClassNotFoundException	исключение может произойти, например, при попытке запуска несуществующего класса с помощью команды java
IOException	Связаны с операциями ввода/вывода, такими как недопустимые входные данные, считывание после конца файла и открытие несуществующего файла.

Ошибки во время выполнения

Ошибки во время выполнения представлены в классе **RuntimeException**, который описывает ошибки программирования, такие как неправильное приведение типов, доступ к массиву вне его границ и числовые ошибки.

Класс	Возникновение исключения
ArithmeticException	Деление целого числа на ноль. Арифметические операции над числами с плавающей точкой не приводят к исключениям
NullPointerException	Попытка доступа к объекту с помощью нулевой переменной ссылочного типа.
IndexOutOfBoundsException	Индекс массива находится вне его границ.
IllegalArgumentException	Методу передается недопустимый или не соответствующий ему аргумент.

Пример использования исключений

В этом примере решения задачи показано объявление, выбрасывание и перехват исключений с помощью изменения метода `setRadius()` класса `Circle`. Метод `setRadius()` будет выбрасывать исключение, если радиус отрицательный.

`RuntimeException`, `Error` и их подклассы называются непроверяемыми (**unchecked**) исключениями. Все остальные исключения называются проверяемыми (**checked**) исключениями, то есть компилятор заставляет программиста проверять и обрабатывать их в блоке `try-catch` или объявлять в заголовке метода.

▼ Класс TestCircleWithPrivateDataFields

```
public class TestCircleWithPrivateDataFields {
  /** Метод main */
  public static void main(String[] args) {
```

```
    // Создать круг с радиусом, равным 5
    Circle myCircle = new Circle(5.0);
    System.out.println("Площадь круга с радиусом "
      + myCircle.getRadius() + " равна " + myCircle.getArea());

    // Увеличить радиус созданного круга на 10%
    myCircle.setRadius(myCircle.getRadius() * 1.1);
    System.out.println("Площадь круга с радиусом "
      + myCircle.getRadius() + " равна " + myCircle.getArea());

    System.out.println("Количество созданных объектов равно "
      + Circle.getNumberOfObjects());
  }
}
```

▼ Класс Circle

```
public class Circle {
  /** Радиус круга */
  private double radius = 1;

  /** Количество созданных объектов */
  private static int numberOfObjects = 0;

  /** Создает круг с радиусом, равным 1 */
  public Circle() {
    numberOfObjects++;
  }

  /** Создает круг с указанным радиусом */
  public Circle(double newRadius) {
    radius = newRadius;
    numberOfObjects++;
  }

  /** Возвращает радиус */
  public double getRadius() {
    return radius;
  }

  /** Присваивает новый радиус */
  public void setRadius(double newRadius) {
    radius = (newRadius >= 0) ? newRadius : 0;
  }

  /** Возвращает количество объектов */
  public static int getNumberOfObjects() {
    return numberOfObjects;
  }
```

```
  /** Возвращает площадь круга */
  public double getArea() {
    return radius * radius * Math.PI;
  }
}
```

В программе `TestCircleWithException` определяется новый класс круга с именем `CircleWithException`, который совпадает с `Circle` из программы `TestCircleWithPrivateDataFields`, за исключением того, что метод `setRadius(double newRadius)` выбрасывает исключение `IllegalArgumentException`, если аргумент `newRadius` отрицателен.

▼ Класс TestCircleWithException

```
public class TestCircleWithException {
  public static void main(String[] args) {
    try {
      CircleWithException c1 = new CircleWithException(5);
      CircleWithException c2 = new CircleWithException(-5);
      CircleWithException c3 = new CircleWithException(0);
    }
    catch (IllegalArgumentException ex) {
      System.out.println(ex);
    }

    System.out.println("Количество созданных объектов: " +
      CircleWithException.getNumberOfObjects());
  }
}
```

▼ Класс CircleWithException

```
public class CircleWithException {
  /** Радиус круга */
  private double radius;

  /** Количество созданных объектов */
  private static int numberOfObjects = 0;

  /** Создает круг с радиусом, равным 1 */
  public CircleWithException() {
    this(1.0);
  }

  /** Создает круг с указанным радиусом */
  public CircleWithException(double newRadius) {
```

```
    setRadius(newRadius);
    numberOfObjects++;
  }

  /** Возвращает радиус */
  public double getRadius() {
    return radius;
  }

  /** Присваивает новый радиус */
  public void setRadius(double newRadius)
      throws IllegalArgumentException {
    if (newRadius >= 0)
      radius = newRadius;
    else
      throw new IllegalArgumentException(
        "Радиус не может быть отрицательным");
  }

  /** Возвращает numberOfObjects */
  public static int getNumberOfObjects() {
    return numberOfObjects;
  }

  /** Возвращает площадь круга */
  public double findArea() {
    return radius * radius * 3.14159;
  }
}
```

Исходный класс `Circle` остался нетронутым, за исключением того, что имя класса изменено на `CircleWithException`, добавлен новый конструктор `CircleWithException(newRadius)`, а метод `setRadius()` теперь объявляет исключение и выбрасывает его, если радиус отрицательный.

Метод `setRadius()` объявляет в своем заголовке о выбрасывании исключения `IllegalArgumentException` (*в классе CircleWithException*). Класс `CircleWithException` будет по-прежнему компилироваться, если часть заголовка `throws IllegalArgumentException` будет удалена из объявления метода, так как этот класс является подклассом `RuntimeException`, и каждый метод может выбросить `RuntimeException` (*непроверяемое исключение*) независимо от того, объявлено ли оно в заголовке метода или нет.

В программе-клиенте создается три объекта типа CircleWithException – c1, c2 и c3 – для проверки обработки исключений. При выполнении `new CircleWithException(-5)` (*в класса TestCircleWithException*) вызывается метод `setRadius()` и выбрасывается исключение `IllegalArgumentException`, так как радиус отрицательный. В блоке `catch`

типом объекта ex является `IllegalArgumentException`, который соответствует объекту исключения, выброшенному методом `setRadius()`, поэтому это исключение перехватывается блоком `catch`. Обработчик исключений отображает в консоли короткое сообщение об исключении с помощью `System.out.println(ex)`. В случае исключения выполнение программы продолжается. Если бы обработчики не перехватили исключение, то программа бы прервалась. Программа-клиент будет по-прежнему компилироваться, если предложение `try` не будет использоваться, так как метод выбрасывает экземпляр класса `IllegalArgumentException`, подкласса `RuntimeException` (*непроверяемое исключение*).

Задание 1

С помощью двух массивов напишите программу, которая предложит пользователю ввести целое число от **1** до **12**, а затем отобразит месяц и количество дней, соответствующие этому целому числу. Если пользователь вводит недопустимое число, то программа должна отображать **Недопустимое** число с помощью перехвата `ArrayIndexOutOfBoundsException`. Программа должна быть реализована таким образом, чтобы предотвратить ввод пользователем **любого числа**, **кроме целого**.

```
String[] months = {"январь", "февраль", "март", "апрель", "май",
  "июнь", "июль", "август", "сентябрь", "октябрь", "ноябрь", "декабрь"};

int[] dom = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

Задание 2

Измените в программе TestLoanClass класс Loan таким образом, чтобы выбросить `IllegalArgumentException`, если годовая процентная ставка, срок или сумма кредита меньше или равны нулю.

Задание 3

Дополните код 1 задания данной практической работы таким образом, чтобы при выборе февраля была возможность записать год. Добавьте в программу метод для расчета является ли год високосным и измените вывод для количества дней в феврале.