

# EXTERNAL JAVA TRAINING TASK SHAPES PART I

---

---

[www.training.by](http://www.training.by)

**Legal Notice:** This document contains privileged and/or confidential information and may not be disclosed, distributed or reproduced without the prior written permission of EPAM Systems.

---

October 8, 2021

## Task Shapes Part I

Необходимо написать приложение и тесты к нему согласно требованиям, приведенным ниже. В приложении должна быть реализована функциональность, определенная индивидуальным заданием.

### Requirements

Пусть задание сформулировано следующим образом: «**Фигура.** Разработать классы Точка и Фигура. Создать методы и тесты: вычисления площади и периметра фигуры; является ли фигура той именно фигурой по определению; пересекает ли фигура только одну из осей координат на заданное расстояние etc.»

- Разработать entity-классы, например: «Разработать классы Точка и Фигура»
- Entity-классы не следует наполнять методами, выполняющими функциональные действия (методами бизнес-логики, такими как вычисление, поиск и т.д.).
- Фигура должна содержать поле id или name.
- Разработать action-классы реализующие заданные функциональности, например: «Реализовать методы вычисления площади и периметра фигуры»
- Параметры, необходимые для создания объектов, читать из файла (.txt). **Часть данных должна быть некорректной.** Если встретилась некорректная строка, приложение должно переходить к следующей строке. Файлы не должны находиться вне каталогов.
- Для чтения информации из файла использовать **только** возможности, появившиеся в Java8.
- Разработать классы-валидаторы для проверки результатов вычислений параметров фигур, а также для валидации исходных данных для создания объектов entity-классов. Например: *корректная* строка в файле для создания объекта **Круг**: «1.0 2.0 3.0», где первое второе - координаты центра, третье - радиус круга; *некорректная* строка в файле для создания объекта **Круг**: «2a.0 3.0 4.1» - недопустимый символ «z», всю строку следует считать некорректной здесь и в случаях приведенных ниже; *некорректная* строка в файле для создания объекта **Круг**: «1.0 2.0» - недостаточно информации для создания объекта (можно использовать значение по умолчанию, н-р: 1); *некорректная* строка в файле для создания объекта **Круг**: «1.0 2.0 -3.0» - невозможно создать Круг с отрицательным радиусом.
- Для классов-сущностей следует **переопределять методы класса Object: toString(), equals(), hashCode()**. Методы класса Objects использовать нельзя.
- При решении задачи для создания entity-классов использовать паттерн Factory Method.
- Все классы приложения должны быть структурированы **по пакетам.**
- Использовать собственные классы исключительных ситуаций.
- **Оформление кода должно соответствовать Java Code Convention.**
- Для записи логов использовать Log4J2 (SLF4J). Логи писать следует в консоль и в файл.
- Код должен быть покрыт Unit-тестами. Использовать TestNG (JUnit). При написании тестов **запрещено**: создавать неаннотированные методы, писать логи и использовать операторы ветвления: if, for, while, do\while, switch; использовать в тест-методе более одного Assert-метода.
- **Обратить внимание на примечания 1 и 2.**

## Индивидуальные задания

- 1 Треугольник.** Разработать классы Точка и Треугольник. Создать методы и тесты: вычисления площади и периметра треугольника; составляют ли точки треугольник (не лежат ли точки на одной прямой); является ли треугольник прямоугольным, равнобедренным, равносторонним, остро/тупоугольным.
- 2 Прямоугольник.** Разработать классы Точка и Прямоугольник. Создать методы и тесты: вычисления площади и периметра фигуры; составляют ли точки прямоугольник (не лежат ли три точки на одной прямой); является ли четырехугольник выпуклым; является ли прямоугольник квадратом, ромбом, трапецией.
- 3 Овал.** Разработать классы Точка и Овал (задан двумя точками описанного прямоугольника). Создать методы и тесты: вычисления площади и периметра фигуры; составляют ли точки овал (не лежат ли две точки на одной прямой, параллельной осям координат); пересекает ли фигура только одну из осей координат на заданное расстояние; является ли фигура овалом, кругом.
- 4 Шар.** Разработать классы Точка и Шар. Создать методы и тесты: вычисления площади поверхности шара, объема шара, соотношения объемов получаемых в результате рассечения шара координатной плоскостью; является ли объект шаром; касается ли шар любой из координатных плоскостей.
- 5 Куб.** Разработать классы Точка и Куб. Создать методы и тесты: вычисления площади поверхности куба, объема куба; соотношения объемов получаемых в результате рассечения куба координатной плоскостью является ли объект кубом; находится ли основание куба на одной из координатных плоскостей.
- 6 Тетраэдр.** Разработать классы Точка и Тетраэдр. Создать методы и тесты: вычисления площади поверхности фигуры и ее объема, а также соотношения объемов получаемых в результате рассечения фигуры координатной плоскостью; является ли объект заданной фигурой; находится ли основание фигуры на одной из координатных плоскостей.
- 7 Пирамида.** Разработать классы Точка и Пирамида. Создать методы и тесты: вычисления площади поверхности фигуры и ее объема, а также соотношения объемов получаемых в результате рассечения фигуры координатной плоскостью; является ли объект заданной фигурой; находится ли основание фигуры на одной из координатных плоскостей.
- 8 Конус.** Разработать классы Точка и Конус. Создать методы и тесты: вычисления площади поверхности фигуры и ее объема, а также соотношения объемов получаемых в результате рассечения фигуры координатной плоскостью; является ли объект заданной фигурой; находится ли основание фигуры на одной из координатных плоскостей.

**Примечание 0:**

Определение фигуры может быть различным. Например, Куб может определяться 8-ю точками или координатой центра и длиной стороны или каким либо еще.

**Примечание 1:**

*Секущие плоскости, фигуры и основания следует ориентировать в пространстве параллельно осям и плоскостям координат, чтобы формулы вычисления сечений и параметров фигур не были слишком сложными.*

**Примечание 2:**

1. После *if* всегда следует положительный сценарий, после *else* - отрицательный
2. Если только *if*, то возможен и отрицательный сценарий
3. В *if*, *for*, *while* обязательно использовать { }
4. Если генерируется *exception*, с помощью *throw*, не ловить его сразу же
5. В *finally* не генерировать исключения и не использовать *return*
6. Не генерировать стандартные исключения. Разрешено только в методах *private*
7. *fileWriter.close()*; - в блок *finally*
8. Регулярные выражения в константы
9. В именах пакетов не использовать большие буквы
10. Не класть сторонние файлы в папки рядом или вместе с классами
11. Размещать файлы только в папках в корне проекта
12. Использовать для файлов только относительные пути. Папка *src* не должна присутствовать в пути к файлу
13. Если константа неизменяемая, то имя должно быть в верхнем регистре, если изменяемая, то как правило именуется как обычное поле класса
14. Элементы перечисления именуются как неизменяемые константы
15. Не увлекаться статическими методами
16. Не объявлять *get*-теры и *set*-теры абстрактными
17. Не давать классам имена, совпадающие с именами стандартных классов и интерфейсов Java !
18. Не разделять объявление переменной и присвоение ей значения в методах, то есть не писать:  
`Integer count;`  
`count = 0;`  
 а надо `Integer count = 0;`
19. Расстояние (в строчках кода) между использованием переменной и ее объявлением должно быть минимально!
20. В одной строчке - одна точка, то есть надо использовать локальные переменные, не надо:  
`sample.getSomething().getData()`  
 надо:  
`Something something = sample.getSomething();`  
`Data data = something.getData();`
21. Не писать `if (isValid == true)`, а писать `if (isValid)`
22. Не писать:  
`if (someValue == EXPECTED_VALUE) {`  
`return true;`  
`} else {`  
`return false;`  
`}`  
 писать:  
`return someValue == EXPECTED_VALUE;`

23. Использовать *assertEquals* вместо *assertTrue(some == other)*
24. Использовать *assertTrue(isValid)* вместо *assertEquals(true, isValid)*
25. Лучше тестовые объекты размещать в тестах в виде констант, а не создавать их в самом методе
26. Не использовать существующий *FactoryMethod* в тестах для создания объектов, объекты в тестах делать через *new*
27. Тест должен иметь структуру: *given*, *when*, *then*, где *given* - прекондишены (инициализация данных), *when* - вызов тестируемого метода (всегда одна строчка), *then* - посткондишн (*assert-метод*)

Ver.	Description of Change	Author	Date	Approved	
				Name	Effective Date
<1.0>	Первая версия	Игорь Блинов	15-09-2016		
<1.1>	update	Игорь Блинов	14-02-2017		
<1.2>	update	Игорь Блинов	09-06-2017		
<2.0>	Вторая версия	Игорь Блинов	02-11-2017		
<2.1>	update	Игорь Блинов	17-01-2018		