На основании формулы (1) с использованием возможностей системы *Wolfram Mathematica*, изобразим графически зависимость сечения рождения пары от энергии фотона в кулоновском поле легких ядер (алюминий *Al* (*Z*=13), кремний *Si* (*Z*=14), фосфор *P* (*Z*=15), рисунок 2, а)) и тяжелых ядер (таллий *Tl* (*Z*=81), свинец *Pb* (*Z*=82), висмут *Bi* (*Z*=83), рисунок 2, б)). При взаимодействии фотона с энергией от 3 до 6,5 МэВ с легкими ядрами вещества сечение образование пары растет медленно (рисунок 2, а)), для тяжелых ядер наблюдается резкий рост сечения, при этом оно принимает достаточно большие значения (рисунок 2, б)). Объясняется это тем, что сечение зависит и от заряда ядра, т.е. рождение электронно-позитронной пары на тяжелых ядрах идет более интенсивно, нежели на легких. Численные значения полученных сечений представлены в виде таблицы 1.

Таблица 1 – Сечения рождения пар на легких и тяжелых ядрах

	Al	*Si*	*P*
$E_\gamma = 3{,}5$ МэВ	0,66368 Фм^2	0,76971 Фм^2	0,88359 Фм^2
$E_\gamma = 4{,}5$ МэВ	8,22530 Фм^2	9,53940 Фм^2	10,95085 Фм^2
$E_\gamma = 5{,}5$ МэВ	14,26314 Фм^2	16,54187 Фм^2	18,98939 Фм^2
$E_\gamma = 6{,}5$ МэВ	19,28952 Фм^2	22,37127 Фм^2	25,68131 Фм^2
	Tl	*Pb*	*Bi*
$E_\gamma = 3{,}5$ МэВ	25,76571 Фм^2	26,40582 Фм^2	27,05379 Фм^2
$E_\gamma = 4{,}5$ МэВ	319,32679 Фм^2	327,26007 Фм^2	335,29069 Фм^2
$E_\gamma = 5{,}5$ МэВ	553,73078 Фм^2	567,48754 Фм^2	581,41310 Фм^2
$E_\gamma = 6{,}5$ МэВ	748,86712 Фм^2	767,47181 Фм^2	786,30477 Фм^2

Заключение. Таким образом, при увеличении энергии фотона и заряда ядра, сечение рождения пар растет, при этом, если энергия слишком мала, то образование пар может не наблюдаться. Образование пар являться основным механизмом, отвечающим за поглощение γ-квантов в области больших энергий. Анализ данных рисунка 2, таблицы 1, а также свойств исследуемых вещества свидетельствует о том, что наилучшей защитой от γ-излучения из них является свинец.

1. Мухин, К.Н. Экспериментальная ядерная физика: Учеб. для вузов. В 2 кн. Кн. 1. Физика атомного ядра. Ч. I. Свойства нуклонов, ядер и радиоактивных излучений / К.Н. Мухин. – М.: Энергоатомиздат, 1993. – 376 с.
2. Батурицкий, М.А. Взаимодействие ионизирующего излучения с веществом: учеб. пособие / М.А. Батурицкий, И.Я. Дубовская. – Минск: РИВШ, 2010. – 220 с.

АРХИТЕКТУРА ПРОГРАММНОГО ТРЕНАЖЕРА РЕШЕНИЯ МАТЕМАТИЧЕСКИХ ЗАДАЧ

Маркова А.А.,
студентка 3 курса ВГУ имени П.М. Машерова, г. Витебск, Республика Беларусь
Научный руководитель – Ермоченко С.А., канд. физ.-мат. наук

Неотъемлемой частью процесса обучения студентов является контроль уровня учебных достижений. В связи со сложностью задач и алгоритмов высшей математики процесс решения задач студентами и процесс проверки результатов преподавателями могут занимать большое количество времени. Актуальность создания программного тренажера обусловлена возможностью автоматизации этих процессов.

Для студентов тренажер предоставит более удобные средства решения задачи заданным методом без необходимости вручную записывать весь процесс решения и все вычисления. Для преподавателя та-

кое средство контроля уровня знаний, умений и навыков студентов позволит сэкономить время, ранее затрачиваемое на проверку письменных работ.

Цель данной работы: спроектировать архитектуру программного обеспечения тренажера, которая позволит гибко расширять приложение, добавляя новые задачи по различным дисциплинам учебных планов.

Материал и методы. Для проверки работоспособности архитектуры необходимо создание некоторых конкретных задач, то в качестве них выберем системы линейных алгебраических уравнений и квадратные уравнения. Материалом в этом случае будут известные алгоритмы решения таких задач. В качестве методов исследования будут использоваться общенаучные методы анализа, сравнения, а также объектно-ориентированный анализ и проектирование.

При проектировании архитектуры тренажера будем использовать объектно-ориентированный язык программирования Java.

Результаты и их обсуждение. Для взаимодействия с алгоритмами решения математических задач была разработана иерархия наследования классов, описывающих задачи. В вершине иерархии находится абстрактный класс `Task`. Данный класс используется для единообразного доступа к методам, отвечающим за выполнение операций над задачами.

Наследниками класса `Task` являются конкретные классы `SLAETask`, предназначенный для хранения информации о системах линейных алгебраических уравнений (СЛАУ), и `SquareEquationTask`, предназначенный для хранения данных о квадратном уравнении.

Генераторы позволяют составить набор неповторяющихся задач определенного типа с готовыми ответами или решениями [2]. Для генерации используется отдельная иерархия наследования с параметризованным интерфейсом `Generator` в вершине иерархии, в котором объявлен параметризованный метод `generate`.

Листинг 1 – Параметризованный интерфейс для генерации задач

```
public interface Generator<T extends Task> {
 T generate();
}
```

Для непосредственной генерации каждого из вариантов решения задачи используется класс, реализующий интерфейс `Generator`. В качестве возвращаемого значения метода `generate` указывается класс конкретной задачи (в случае СЛАУ – `SLAETask`).

Чтобы можно было использовать иерархию классов, отвечающих за хранение информации о задачах, воспользуемся возможностями ретроспекции языка программирования Java (Reflection API) [1].

Класс `CreatorGenerator` помогает получить объект класса для генерации конкретной задачи, работая лишь на уровне абстракций, используя информацию о переданном объекте класса.

Листинг 2 – Применение ретроспекции для использования задач и генераторов

```
public class CreatorGenerator {
 private static Map<Class<? extends Task>,
 List<Generator<? extends Task>>> generators;
 static { /* заполнение списка генераторов */ }
 public static <T extends Task>
                  Generator<T> create (Class<T> taskClass) {
 /* выбор случайного генератора */ }
}
```

Чтобы сгенерировать конкретную задачу с использованием созданного набора классов, необходимо у класса `CreatorGenerator` вызвать статический метод `create`, передав в него в качестве параметра класс требуемой задачи.

Листинг 3 – Создание объекта класса SLAETask с использованием генератора

```
SLAETask t = CreatorGenerator.create(
                          SLAETask.class).generate();
```

Процесс решения задачи состоит из последовательных шагов, определяемых конкретным алгоритмом. Для реализации шагов решения была разработана иерархия наследования классов с интерфейсом `SolverStep` в вершине иерархии. Этот интерфейс определяет метод `doStep`, который принимает в качестве входных параметров конкретную задачу и данные для выполнения шага.

Листинг 4 – Интерфейс SolverStep для выполнения шага задачи

```
public interface SolverStep<T extends Task> {
 void doStep(T task, MethodsData data);
}
```

Сохранение последовательности шагов, выполнявшихся студентом при решении, реализует класс `Solver`. Этот класс содержит список, состоящий из названия шага и данных для его выполнения, заполненных студентом.

Для демонстрации работоспособности разработанной архитектуры программного тренажера было создано web-приложение для решения СЛАУ. При загрузке страницы с выбранным методом (на рисунке 1 метод Гаусса решения СЛАУ) вызывается генератор СЛАУ. Студенту в виде списка предоставляются доступные шаги. При выборе конкретного шага на странице отобразится набор полей для заполнения данными, необходимыми для его выполнения.

При нажатии на кнопку «Выполнить» заполненные данные отправляются на сервер, где производятся вычисления, и СЛАУ обновляется.

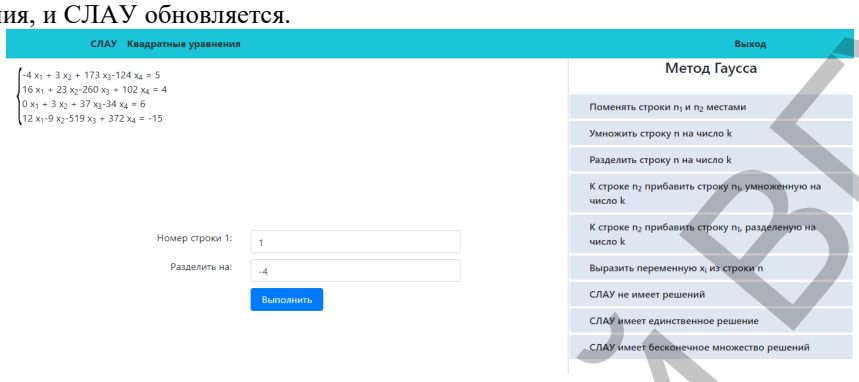

Рисунок 1 – Выполнение шага для решения задачи в web-приложении

Заключение. Такие мощные инструменты объектно-ориентированного языка Java как обобщения, интроспекция, рефлексия, позволяют динамически модифицировать программу во время выполнения, что предоставляет новые возможности при написании кода.

В результате была спроектирована архитектура программного обеспечения тренажера, которая позволяет гибко расширять приложения, добавляя новые задачи по различным дисциплинам учебных планов, проанализировать выполненные студентом действия и оценить уровень его знаний, умений и навыков.

1. Гослинг, Дж., Джой, Б., Стил, Г. Л., Брача, Г., Бакли, А. Язык программирования Java SE 8. Подробное описание. – Москва: Вильямс, 2015. – 672 с.
2. Коновалов, Я.Ю. Генератор контрольных заданий по высшей математике: опыт создания и применения / Я.Ю. Коновалов, С.К. Соболев // ФГБОУ ВПО «МГТУ им. Н.Э.Баумана». – 2015. – №4. – С. 1046-1055.

ИСПОЛЬЗОВАНИЕ ТЕОРИИ ГРУПП ТОЧЕК НА ЭЛЛИПТИЧЕСКОЙ КРИВОЙ ДЛЯ СОЗДАНИЯ ЭЛЕКТРОННОЙ ПОДПИСИ

Марчук К.С.,

студент 2 курса БГТУ, г. Минск, Республика Беларусь

Научный руководитель – Асмыкович И.К., канд. физ.-мат. наук, доцент

Долгое время люди копили информацию. Сначала это были рисунки на стенах, после – печатный или письменный текст, а в наше время мы имеем огромное количество информации в электронном виде. Ни для кого не секрет, что большая часть информации имеет ценность, особенно для юридических лиц. В нашем примере, информация нуждается в проверке принадлежности – подписи.

ECDSA (Elliptic Curve Digital Signature Algorithm) – криптографический алгоритм с открытым ключом для создания цифровой подписи, определённый в группе точек эллиптической кривой [1]. Подпись создается секретно, но может быть публично проверена. Это означает, что только один субъект может создать подпись сообщения, но любой может проверить её корректность.

До того, как ECC стала популярной, почти все алгоритмы с открытым ключом основывались на RSA, DSA и DH [2], альтернативных криптосистемах на основе модулярной арифметики. Они по-прежнему популярны, и часто используются вместе с ECC. Однако основы ECC всё ещё являются для большинства людей загадкой.

Цель – рассмотреть конкретный пример работы алгоритма подписи данных, основанного на теории эллиптических кривых.

Материал и методы.В Биткойне [3] используется вариант эллиптической криптографии secp256k1. Покажем, откуда берутся секретные и открытые ключи и как они связаны друг с другом. В ECDSA секретный ключ – это случайное целое число между 1 и значением порядка – количеством элементов группы.Открытый же ключ получается из секретного при помощи операции скалярного ум-