**Министерство образования и науки Кыргызской Республики**

**Кыргызский Государственный Технический Университет**

**им. И. Раззакова**

**Кафедра программного обеспечения компьютерных систем**

**НАПРАВЛЕНИЕ—710400 «ПРОГРАММНАЯ ИНЖЕНЕРИЯ»**

**Дисциплина «Объектно-ориентированное проектирование и программирование»**

**Пояснительная записка**

**к курсовой работе студента**

Выполнил: Именов Алишер

Группа: ПИ-ИСОП-1-21

Проверил: Мусабаев Э.Б.

Бишкек 2024

Оглавление

[Введение 3](#_Toc167557875)

[ГЛАВА 1. АНАЛИТИЧЕСКАЯ ЧАСТЬ 3](#_Toc167557876)

[Актуальность темы 3](#_Toc167557877)

[Цель работы 3](#_Toc167557878)

[Задачи 3](#_Toc167557879)

[Объект и предмет исследования 3](#_Toc167557880)

[Обзор литературы по теме исследования 3](#_Toc167557881)

[Основные теоретические положения и концепции 3](#_Toc167557882)

[Требования 3](#_Toc167557883)

[функциональные требования 4](#_Toc167557884)

[Основные функции: 4](#_Toc167557885)

[Дополнительные функции: 4](#_Toc167557886)

[Требования к интерфейсу 4](#_Toc167557887)

[Требования к производительности 4](#_Toc167557888)

[Глава 2. Практическая часть 4](#_Toc167557889)

[UML 4](#_Toc167557890)

[Диаграмма классов 4](#_Toc167557891)

[Заключение 9](#_Toc167557892)

[Краткое изложение основных результатов работы 9](#_Toc167557893)

[Выводы по проведенному исследованию 9](#_Toc167557894)

[Приложение 9](#_Toc167557895)

[Главный класс: 9](#_Toc167557896)

# Введение

**Тамагочи** – это виртуальный питомец, который требует ухода и внимания пользователя. Основная задача игрока – заботиться о питомце, вовремя кормить его, играть с ним и следить за его здоровьем.

# ГЛАВА 1. АНАЛИТИЧЕСКАЯ ЧАСТЬ

Актуальность темы**:** Виртуальные питомцы, такие как Тамагочи, стали популярными в конце 1990-х годов и остаются интересными для изучения с точки зрения их влияния на развитие цифровых развлечений и взаимодействия пользователя с программами.

Цель работы**:** Исследование и разработка программы виртуального питомца на примере Тамагочи.

Задачи**:**

* Провести обзор литературы по теме виртуальных питомцев.
* Изучить основные теоретические концепции, лежащие в основе создания виртуальных питомцев.
* Разработать программу виртуального питомца.
* Провести тестирование программы и проанализировать результаты.

Объект и предмет исследования**:**

Объектом исследования является виртуальный питомец Тамагочи, предметом исследования – методы и технологии разработки программного обеспечения для создания виртуальных питомцев.

Обзор литературы по теме исследования**:** Виртуальные питомцы появились в 1990-х годах и стали популярными благодаря компактным электронным устройствам. Исследования показывают, что взаимодействие с виртуальными питомцами может оказывать положительное влияние на развитие ответственных и заботливых качеств у детей.

Основные теоретические положения и концепции**:** Основные концепции включают в себя теорию взаимодействия человека с компьютером, основы программирования и принципы создания интерактивных приложений.

## Требования

1. Реализовать минимум 10 классов, включая наследование.
2. Использовать перегрузку методов и шаблоны.
3. Переопределить операции, где это применимо.
4. Обработать исключения.
5. Включить использование виртуальных методов и статических полей.
6. Отправить код на GitHub или GitLab.
7. Подготовить отчет.

## функциональные требования

### Основные функции:

* + Кормление питомца.
  + Игра с питомцем.
  + Проверка состояния питомца (уровень голода и счастья).
  + Влияние времени на состояние питомца (уменьшение уровня голода и счастья со временем).
  + Чистка питомца

### Дополнительные функции:

* + Возможность ввода имени питомца.
  + Сохранение и загрузка состояния питомца.
  + Награда за игру
  + Возможность покупки еды, медикаментов, игрушек

### Требования к интерфейсу

* Интерфейс должен быть текстовым, с возможностью выбора действий через консольное меню.
* Программа должна запрашивать имя питомца при запуске.
* Меню должно включать пункты для кормления, игры, проверки состояния и выхода из программы.

### Требования к производительности

* Программа должна работать без задержек и сбоев на стандартных ПК.
* Время отклика на действия пользователя не должно превышать 1 секунды.

# Глава 2. Практическая часть

## UML

### Диаграмма классов

#### +-------------------+

#### | Attribute |

#### +-------------------+

#### | - value: int |

#### | - min\_value: int |

#### | - max\_value: int |

#### +-------------------+

#### | + increase(int) |

#### | + decrease(int) |

#### | + get(): int |

#### +-------------------+

#### ^

#### |

#### +--------------------+

#### | |

#### +-------------------+ +-------------------+

#### | Health | | Hunger |

#### +-------------------+ +-------------------+

#### | - value: int | | - value: int |

#### | - min\_value: int | | - min\_value: int |

#### | - max\_value: int | | - max\_value: int |

#### +-------------------+ +-------------------+

#### | + increase(int) | | + increase(int) |

#### | + decrease(int) | | + decrease(int) |

#### | + get(): int | | + get(): int |

#### +-------------------+ +-------------------+

#### ^ ^

#### | |

#### +-------------------+--------------------+

#### | |

#### +-------------------+ +-------------------+ +-------------------+

#### | Happiness | | Energy | | Cleanliness |

#### +-------------------+ +-------------------+ +-------------------+

#### | - value: int | | - value: int | | - value: int |

#### | - min\_value: int | | - min\_value: int | | - min\_value: int |

#### | - max\_value: int | | - max\_value: int | | - max\_value: int |

#### +-------------------+ +-------------------+ +-------------------+

#### | + increase(int) | | + increase(int) | | + increase(int) |

#### | + decrease(int) | | + decrease(int) | | + decrease(int) |

#### | + get(): int | | + get(): int | | + get(): int |

#### +-------------------+ +-------------------+ +-------------------+

#### +-------------------+ +-------------------+

#### | Age | | SaveLoad |

#### +-------------------+ +-------------------+

#### | - age: int | | |

#### +-------------------+ +-------------------+

#### | + increase(): void| | + saveToFile(...) |

#### | + get(): int | | + loadFromFile(...) |

#### +-------------------+ +-------------------+

#### +----------------------------+

#### | RandomEvent |

#### +----------------------------+

#### | + triggerEvent(...) |

#### +----------------------------+

#### ^

#### |

#### +--------------------+

#### |

#### +-------------------+

#### | SimpleRandomEvent |

#### +-------------------+

#### | |

#### +-------------------+

#### +----------------------------+

#### | MiniGame<T> |

#### +----------------------------+

#### | + play(attribute: T, |

#### | money: int&): void |

#### +----------------------------+

#### ^

#### |

#### +--------------------+

#### | |

#### +-------------------+ +-------------------+

#### | GuessingGame | | ArithmeticGame |

#### +-------------------+ +-------------------+

#### | | | |

#### +-------------------+ +-------------------+

#### +-------------------+ +-------------------+

#### | Economy | | Interaction |

#### +-------------------+ +-------------------+

#### | - money: int | | |

#### +-------------------+ +-------------------+

#### | + getMoney(): int | | + showOptions(): |

#### | + earnMoney(...) | | + getUserChoice():|

#### | + spendMoney(...) | | + handleChoice(...)|

#### | + buyFood(...) | | + showShopOptions()|

#### | + buyMedicine(...)| | + handleShopChoice()|

#### | + buyToy(...) | +-------------------+

#### +-------------------+

#### ^

#### |

#### +--------------------+

#### |

#### +-------------------+

#### | Tamagotchi |

#### +-------------------+

#### | - hunger: Hunger |

#### | - happiness: Happiness |

#### | - energy: Energy |

#### | - cleanliness: Cleanliness |

#### | - age: Age |

#### | - health: Health |

#### | - economy: Economy|

#### | - interaction: Interaction |

#### | - saveLoad: SaveLoad |

#### | - randomEvent: RandomEvent |

#### +-------------------+

#### | + run(): void |

#### | + isAlive(): bool |

#### +-------------------+

#### Проверка на соответствие требованиям

10 классов, включая наследники

 **Attribute** (базовый класс)

* Health (наследует Attribute)
* Hunger (наследует Attribute)
* Happiness (наследует Attribute)
* Energy (наследует Attribute)
* Cleanliness (наследует Attribute)

 **Age**

 **SaveLoad**

 **RandomEvent** (базовый класс)

* SimpleRandomEvent (наследует RandomEvent)

 **MiniGame<T>** (шаблонный базовый класс)

* GuessingGame (наследует MiniGame<Attribute>)
* ArithmeticGame (наследует MiniGame<Attribute>)

 **Economy**

 **Interaction**

 **Tamagotchi**

**Виртуальные методы, перегрузка методов и шаблоны**

* Виртуальные методы реализованы в RandomEvent, MiniGame и его наследниках.
* Перегрузка методов реализована в классе Attribute.
* Шаблонный класс MiniGame реализует механизм мини-игр.

**Исключения**

* Исключения обработаны в классе SaveLoad и Tamagotchi.

**Статические поля**

* Статическое поле instanceCount в классе Tamagotchi.

# Заключение

Краткое изложение основных результатов работы**:** В результате проведенного исследования была разработана и протестирована программа виртуального питомца. Программа демонстрирует базовые возможности взаимодействия пользователя с виртуальным питомцем, включая кормление, игры и мониторинг состояния питомца.

Выводы по проведенному исследованию**:** Разработка программного обеспечения для виртуальных питомцев требует учета различных аспектов взаимодействия пользователя с программой. Полученные результаты подтверждают возможность создания интерактивных приложений, способных моделировать поведение виртуальных питомцев.

# Приложение

## Главный класс:

class Tamagotchi {

private:

std::string name;

Attribute hunger{ 50, 0, 100 };

Attribute happiness{ 50, 0, 100 };

Attribute energy{ 50, 0, 100 };

Attribute cleanliness{ 50, 0, 100 };

Age age;

Health health;

SaveLoad saveLoad;

SimpleRandomEvent randomEvent;

Interaction interaction;

Economy economy;

static int instanceCount;

public:

Tamagotchi() { instanceCount++; }

~Tamagotchi() { instanceCount--; }

static int getInstanceCount() { return instanceCount; }

void setName(const std::string& newName) {

name = newName;

}

const std::string& getName() const {

return name;

}

void run() {

srand(static\_cast<unsigned>(time(0)));

std::vector<std::reference\_wrapper<Attribute>> attributes = { hunger, happiness, energy, cleanliness };

try {

int econ = economy.getMoney();

saveLoad.loadFromFile("tamagotchi\_status.txt", attributes, age, econ, name);

}

catch (const TamagotchiException& e) {

std::cerr << e.what() << "\n";

}

std::cout << "Welcome to Tamagotchi, " << name << "!\n";

char choice;

do {

interaction.showOptions();

choice = interaction.getUserChoice();

interaction.handleChoice(choice, hunger, happiness, energy, cleanliness, economy);

if (isAlive()) {

randomEvent.triggerEvent(hunger, happiness, energy, cleanliness);

age.increase();

std::cout << "Age: " << age.get() << "\n";

}

else {

std::cout << "Your Tamagotchi has passed away. RIP.\n";

break;

}

} while (choice != '7');

try {

saveLoad.saveToFile("tamagotchi\_status.txt", attributes, age, economy.getMoney(), name);

}

catch (const TamagotchiException& e) {

std::cerr << e.what() << "\n";

}

}

bool isAlive() const {

return hunger.get() > 0 && happiness.get() > 0 && energy.get() > 0 && cleanliness.get() > 0;

}

};

Главное меню приложения:

