# введение

Целью учебной практики по профессиональному модулю ПМ.01 «Разработка, администрирование и защита баз данных» является углубление знаний и практических умений и получение начального практического опыта по основному виду деятельности «Разработка модулей программного обеспечения для компьютерных систем» и овладение соответствующими общими и профессиональными компетенциями: ОК 1, ОК 2, ОК 3, ОК 4, ОК 5, ОК 6, ОК 7, ОК 8, ОК 9, ОК 10, ОК 11, ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5., ПК 1.6.

Учебная практика проводится на базе факультета СПО Университета ИТМО.

Результатом практики является разработка консольного приложения на языке Java с использованием коллекций.

# практическое задание

- создать описанную нижу струтуру

- заполнить тестовыми сущностями

- продемострировать корректную работу кода

===стуктура классов===

GameServer

- World world //мир c сущностями

- int updateCounter //колчество обновлений сервера

- public void updateServer()

class World

- List<Entity> entities

- public void updateWorld()

- public List<Entity> getEntitiesInRegion(int x, int z, double range)

- public List<Entity> getEntitiesNearEntity(Entity entity, double range)

- public List<EntityGuard> getGuardiansInRegion(int x, int z, double range)

abstract class Entity

- String title

- int xPos

- int zPos

- int age //количество пережитых обновлений

- double maxHealth //максимальное колво хп

- double health //текущее количество хп

- public void update() //age++

- public boolean attackEntityFrom(Entity entity, double damage)

class EntityPlayer extends Entity

- int exp

- public void update()

- public boolean attackEntityFrom(Entity entity, double damage)

- public double calculateDamage()

class EntityGuard extends Entity

- public void update()

class EntityMonster extends Entity

- double damage

- public void update()

===логика работы===

GameServer

updateServer фунция сначала увеличивает updateCounter на 1

потом вызывает у world функцию updateWorld

World

метод updateWorld перебирает всех живых сущностей и вызывает у них функцию update

после этого в конце обновления (мира) из списка должны удаляться все сущности у которых health <= 0

методы getEntities(InRegion/getNearEntity) принимают координаты и радиус/сущность (взять ее координаты)

и возвращают список сущностей в радиусе от этих координат

метод getGuardiansInRegion делает то же самое, но ищет только сущностей EntityGuard

Entity описывает вообще всех сущностей

метод update в нем должен только увеличивать поле age на 1

метод attackEntityFrom принимает атакующую сущность и урон

должен вычитать урон из жизней сущности, возвращает true, если сущность умерла (health <= 0)

EntityGuard наследует Entity

метод update сначала должен вызывать родительскую реализацию

после чего искать ближайшего сущность EntityPlayer и двигаться к ней

EntityPlayer наследует Entity

метод update сначала должен вызывать родительскую реализацию

также раз в 2 обновления если health < maxHealth && health > 0, health нужно увеличить на 1

метод attackEntityFrom переопределяет родительский метод: если в радиусе 2 есть живые EntityGuard

то они должны принимать урон вместо игрока (вызывать метод attackEntityFrom для них)

если рядом нет EntityGuard и после удара игрок выживает он вызывает attackEntityFrom для атакующей сущности

метод calculateDamage расчитывает урон игрока по формуле 3 + exp / 2

если игрок убивает сущность, то exp увеличивается на 1

EntityMonster наследует Entity

метод update сначала должен вызывать родительскую реализацию

после чего искать ближайшего сущность EntityPlayer и двигаться к ней

если в радиусе 2 есть живая(ые) сущность(ти) EntityPlayer

для ближайшего из них необходимо вызывать метод attackEntityFrom(this)

===особенности===

\*движение сущностей

сущность за 1 обновление может смещаться на 1 по xPos и на 1 по zPos

\*убийство

когда сущность убивает другую сущность об этом необходимо вывести уведомление в консоль (кто и кого убил)

\*общие требования к коду

если класс наследуется другим классом, все его поля должны быть protected, если нет - private

у всех классов должен быть переопределен метод toString() и должны быть геттеры и сеттеры

===важно===

вы можете добавлять свои классы, поля и методы для более удобного и "правильного" выполнения работы

# выполнение



*Код программы.*



*Первые 3 из 20 итераций программы.*

# заключение

В результате выполнения учебной практики разработано консольное приложение на языке Java с использованием коллекций.

В ходе практики:

* Была создана система классов;
* были реализованы метода, требуемые по заданию;
* было проведено тестирование работы программы.

Все задачи, поставленные на практику, выполнены.