Министерство образования Республики Беларусь
Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Лабораторная работа № 1

по дисциплине
«Современные языки программирования»
«Использование языка программирования Kotlin»
Вариант № 27

Преподаватель:
Усенко Ф. В.
Выполнил:
Стрижевский М. В.
Группы 310902

Минск БГУИР 2024

Задание

Добавить монстров Кабан и Оборотень. Организовать метод убийства монстра. Переделать работу классов так, чтобы монстры сообщали квесту о своём убийстве.

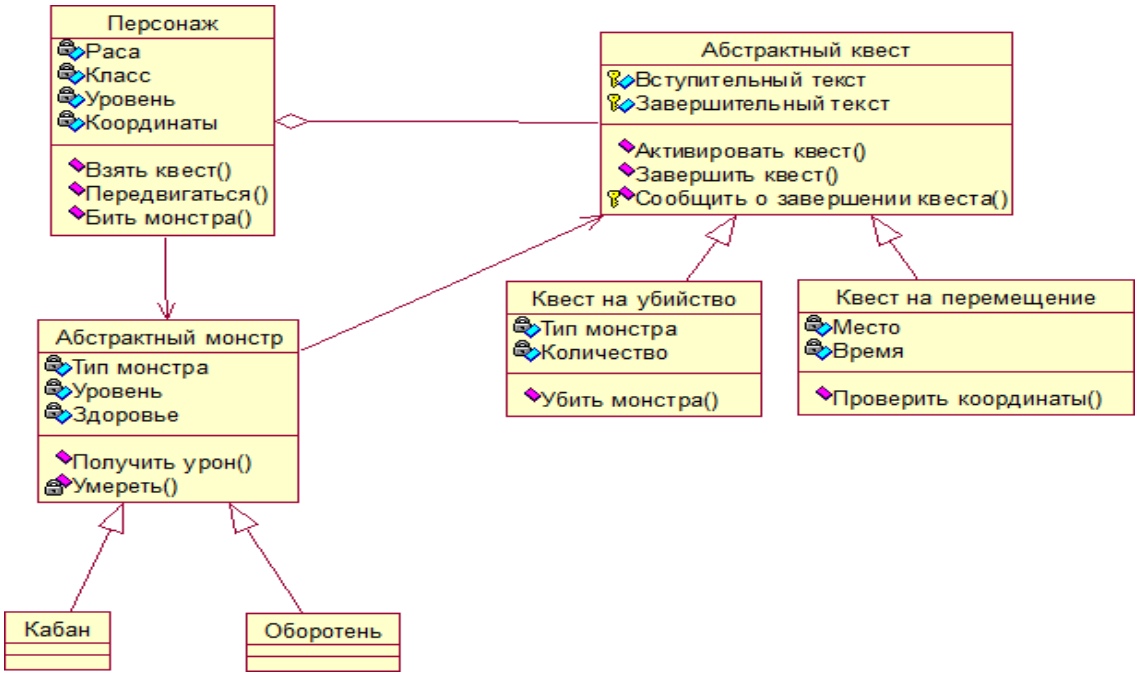

Рисунок 1 – Диаграмма классов

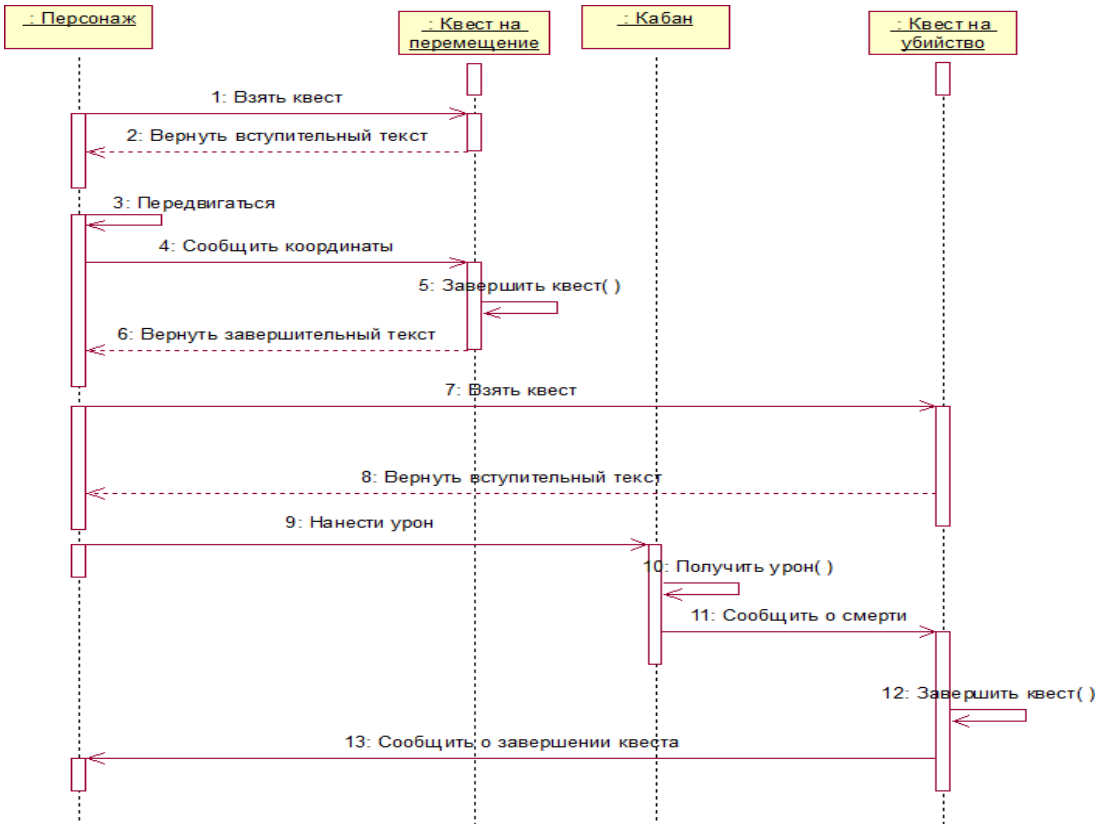

Рисунок 2 – Диаграмма последовательности

Код программы

```
fun main() {
    val player = Player(x = 0,y = 0)
    println("q - quite\nm <x> <y> - move to x,y\nd - damage\nt - take quest")
    while (true) {
        var input = readln()
        when {
            input.equals("q") -> break
            input.equals("m") -> player.move(readln().toInt(), readln().toInt())
            input.equals("d") -> player.hurmMonster()
            input.equals("t") -> player.takeQuest()
        }
    }
}


class Player(
    private val species : String = "human",
    private var plClass : String = "warrior",
    private var level : Int = 1,
    private var x : Int,
    private var y : Int
)
{
    val quests = arrayOf<Quest>(
        MoveQuest(5,5,0),
        KillQuest(EnemyType.BOAR,2))
    var myQuest : Quest? = null;

    fun takeQuest() {
        if (myQuest?.isCompleted() == false) {
            println("Current quest not completed")
            return
        }
        for (q in quests) {
            if (q.isAccepted()) continue
            println(q.getStartText())
            println("Accept (y/n): ")
            if (readln()[0] == 'y'){
```

```
                myQuest = q
                q.accept(this)
                break
            }
        }
    }

    fun move(newX : Int, newY : Int) {
        x = newX;
        y = newY;
        if (myQuest is MoveQuest) {
            (myQuest as MoveQuest).checkPosition(this);
        }
    }

    fun hurmMonster() {
        if (myQuest is KillQuest) (myQuest as KillQuest).hurmMonster(level);
    }

    fun getX() : Int = x
    fun getY() : Int = y

}

abstract class Quest (
        private val startText : String,
    private val complText : String,
    protected var accepted : Boolean = false,
    protected var completed : Boolean = false
){

    fun getStartText() : String = startText

    fun getComplText() : String = complText

    open fun getStateText(player : Player) : String = "Quest accepted"

    fun accept(player : Player) {
        accepted = true
        println(getStateText(player))
```

```
    }
    fun complete() {
        if (accepted) completed = true
        println(getComplText());
    }

    fun isAccepted() : Boolean = accepted
    fun isCompleted() : Boolean = completed

}

class KillQuest : Quest {

    private val enemyType : EnemyType
    private val count : Int
    private var killed : Int = 0
    var enemies : Array<Enemy>

    constructor(enemyType : EnemyType,
                count : Int
            ) : super("There are many monsters in the western forests. Kill
$count*$enemyType",
                "Quest completed") {
        this.enemyType = enemyType
        this.count = count
        enemies = Array(count){
            when(enemyType){
                EnemyType.BOAR -> Boar(1)
                EnemyType.WEREWOLF -> Werewolf(1)
            }
        }
    }

    override fun getStateText(player : Player) : String =
            "${super.getStateText(player)}\nRequested: $count*$enemyType\nKilled:
$killed"

    fun hurmMonster(damage : Int) {

        for (e in enemies) {
```

```
            if (e.isAlive()) {
                e.takeDamage(damage)
                if (e.isAlive() == false) killed++
                break;
            }
        }
        if (isCompleted() == false && killed == count) complete()
    }

}

class MoveQuest(
    private val x : Int,
    private val y : Int,
    private val t : Int
) : Quest("Something strange appeared in the distance. Check it",
        "Quest completed") {

    override fun getStateText(player : Player) : String =
        "${super.getStateText(player)}\nDestination: x=$x, y=$y\nCurrent location:
x=${player.getX()}, y=${player.getY()}"

    fun checkPosition(player : Player) {
        if (isCompleted() == false && player.getX() == x && player.getY() == y)
complete()
    }

}

enum class EnemyType() {
    BOAR,
    WEREWOLF
}

abstract class Enemy(
    private val enemyType : EnemyType,
    private val level : Int,
    private var health : Int
) {
    fun takeDamage(damage : Int) {
```

```
        health -= damage;
        println("$enemyType took damage")
        if (health <= 0) die()
    }

    private fun die() {
                if (health < 0) health = 0
                println("$enemyType died")
    }

    fun isAlive() : Boolean = health != 0
}

class Boar(level : Int) : Enemy(EnemyType.BOAR, level, 1*level)
class Werewolf(level : Int) : Enemy(EnemyType.WEREWOLF, level, 2*level)
```

Результат выполнения программы представлен на рисунке 1.

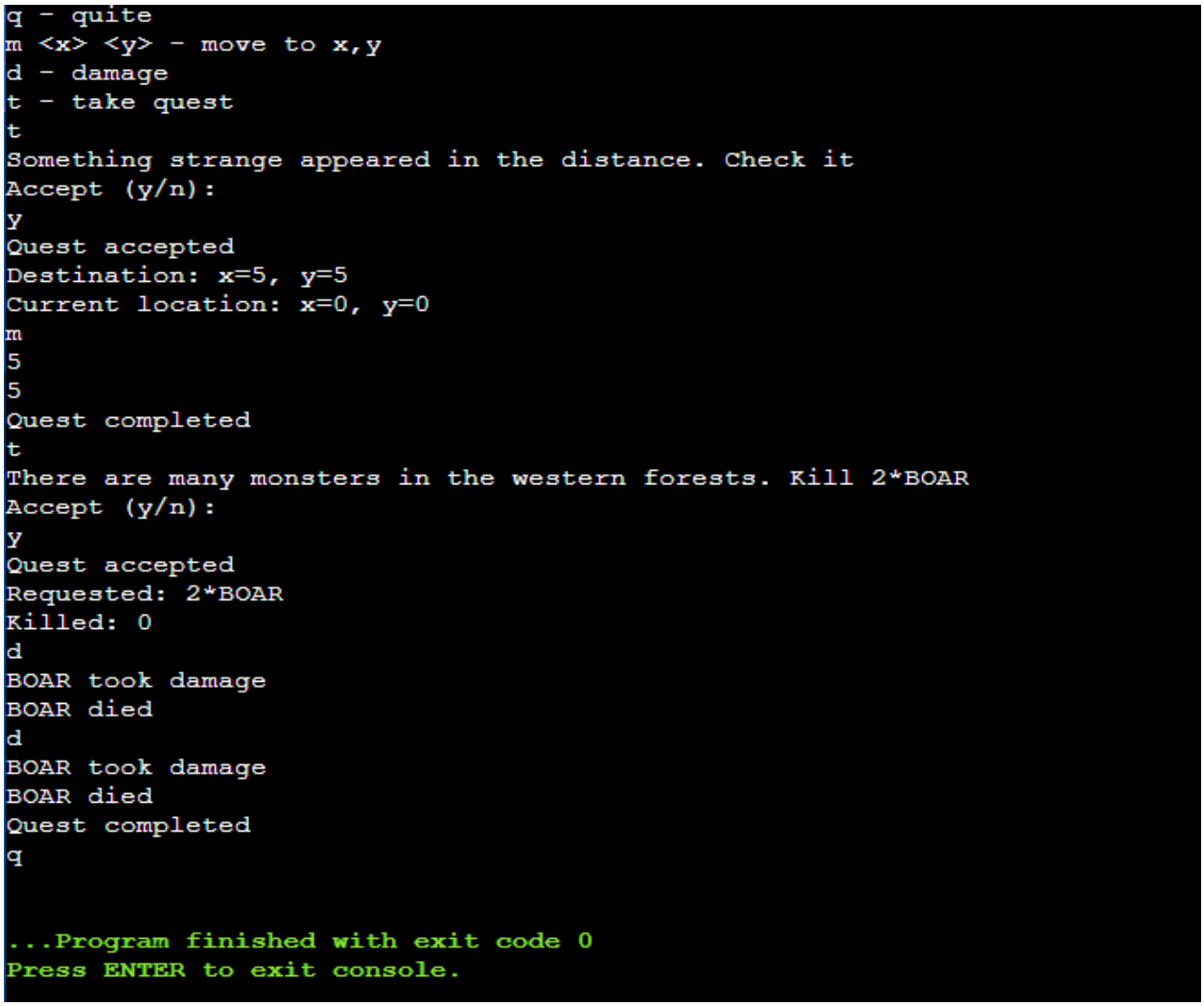

Рисунок 1