Комбінований іспит

з дисципліни «Технології об’єктно-орієнтованого програмування»

студента групи ІТУ-19-2

Куренкова Богдана Михайловича

Екзаменаційний білет № 18

1. Константи у класі. Константні об’єкти

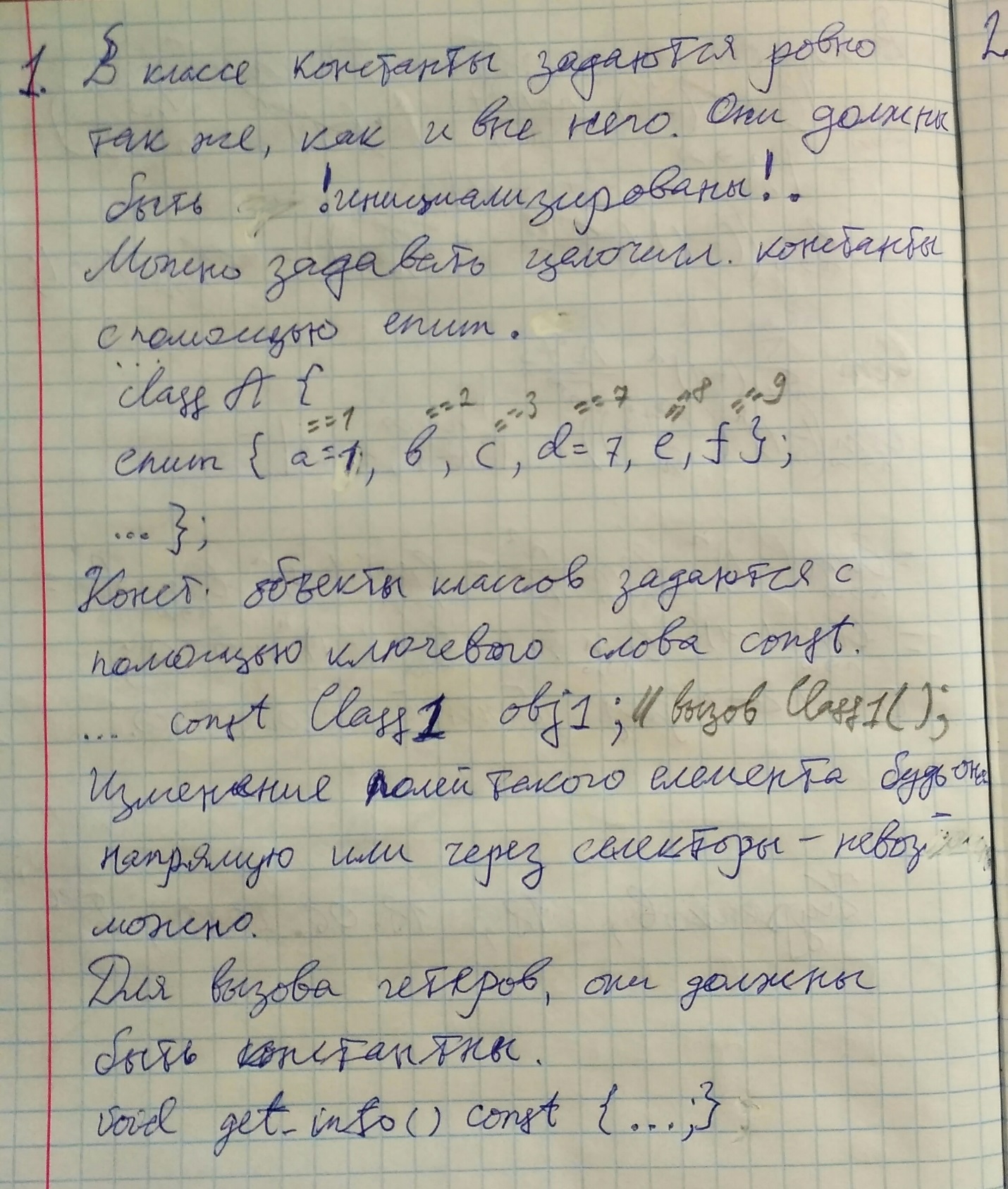
2. Чисто віртуальні функції та абстрактні базові класи

3. Практичне завдання

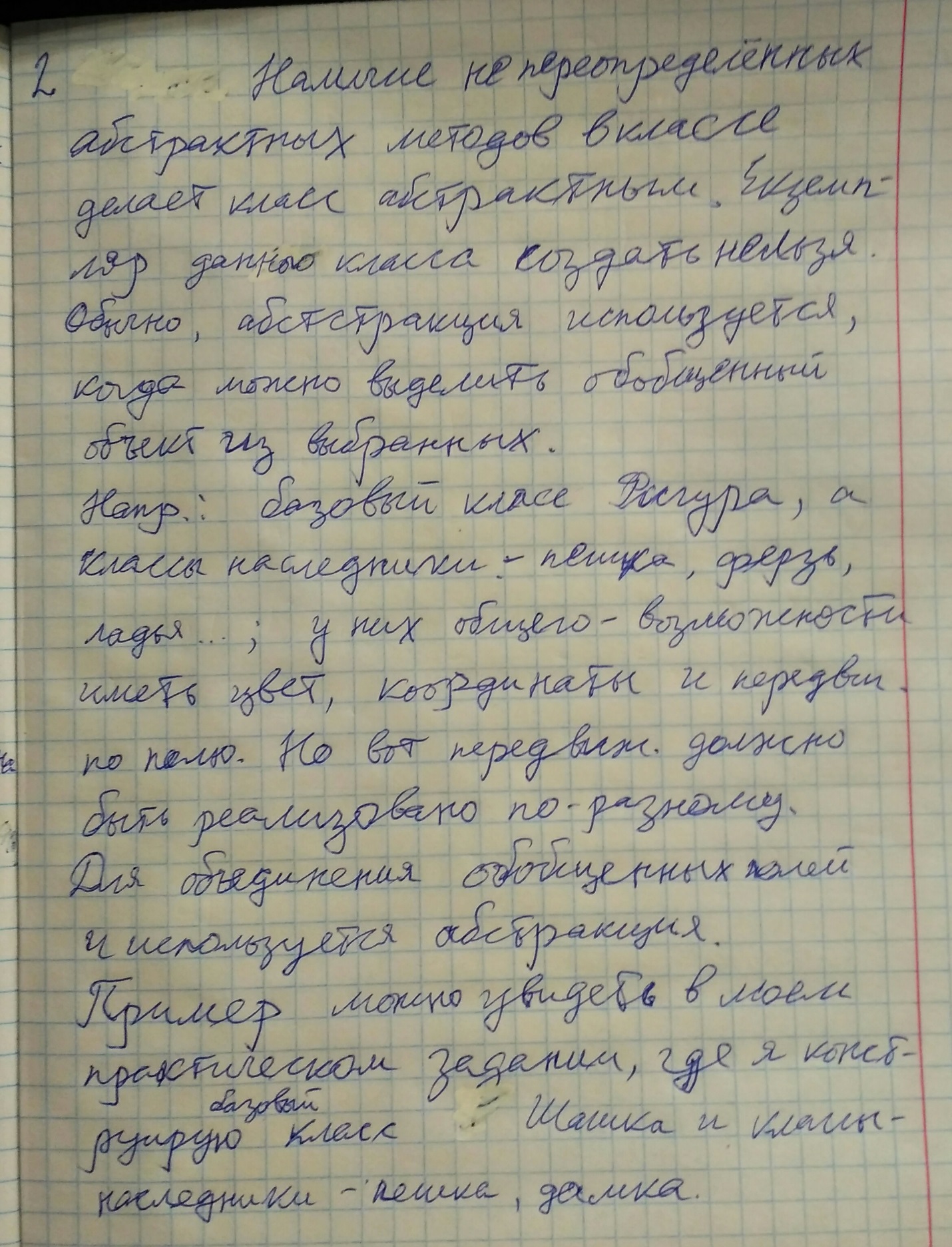
Реалізувати абстрактний базовий клас Шашка, що містить колір та координати на дошці. Чисто віртуальні функції – виведення інформації про шашку та перевірку можливості ходу в задану позицію. Реалізувати похідні класи - звичайна шашка та дамка. Створити клас-контейнер Дошка, що містить динамічний список покажчиків на базовий клас. Реалізувати методи додавання шашок з контролем можливості встановлення на певну клітину. Реалізувати методи класу контейнеру: виведення інформації про всі шашки на дошці, перевірка на можливість ходу у задану клітину для кожної шашки.

Продемонструвати роботу всіх методів класу-контейнеру.

1. **Константи у класі. Константні об’єкти**



**2. Чисто віртуальні функції та абстрактні базові класи**



**3. Практичне завдання**

#include <iostream>

class Chesspiece {

private:

Chesspiece() = default;

protected:

bool color;

int x;

int y;

public:

Chesspiece(bool color, int x, int y) : color(color), x(x), y(y) {}

virtual void get\_info() = 0;

virtual bool moveable(int x, int y) = 0;

int get\_x() { return x; }

int get\_y() { return y; }

bool get\_color() { return color; }

};

class Checker : public Chesspiece {

Checker() = default;

public:

Checker(bool color, int x, int y) : Chesspiece(color, x, y) {}

void get\_info() override {

color ? std::cout << "Белая " : std::cout << "Черная ";

std::cout << "пешка на позиции (" << x << "," << y << ")\n";

}

bool moveable(int x, int y) override {

return (this->x == x && abs(this->y - y));

}

};

class King : public Chesspiece {

King() = default;

public:

King(bool color, int x, int y) : Chesspiece(color, x, y) {}

void get\_info() override {

color ? std::cout << "Белая " : std::cout << "Черная ";

std::cout << "ферзь на позиции (" << x << "," << y << ")\n";

}

bool moveable(int x, int y) override {

return (abs((this->x - x) == abs(this->y - y)) && (this->x != x) || ((this->x == x) && (this->y != y)) || ((this->x != x) && (this->y == y)));

}

};

struct Elem {

Chesspiece\* item;

Elem\* next;

Elem() { item = nullptr; next = nullptr; }

};

class Board {

Elem\* head;

public:

Board() : head(nullptr) {}

void Add\_chess(Chesspiece\* elem) {

if (elem->get\_x() > 0 && elem->get\_x() < 9 && elem->get\_y() > 0 && elem->get\_y() < 9) {

Elem\* tmp = new Elem;

if (!strcmp(typeid(\*elem).name(), "class Checker"))

tmp->item = new Checker(elem->get\_color(), elem->get\_x(), elem->get\_y());

else

tmp->item = new King(elem->get\_color(), elem->get\_x(), elem->get\_y());

tmp->item = elem;

tmp->next = head;

head = tmp;

}

}

void Add\_chess(Chesspiece\* elem, int x, int y) {

Elem\* tmp = new Elem;

if (!strcmp(typeid(\*elem).name(), "class Checker"))

tmp->item = new Checker(elem->get\_color(), x, y);

else

tmp->item = new King(elem->get\_color(), x, y);

tmp->item = elem;

tmp->next = head;

head = tmp;

}

void Delete\_chess() {

Elem\* tmp = head;

head = head->next;

tmp->next = nullptr;

delete tmp;

}

void get\_info() {

std::cout << "Доска содержит:\n";

Elem\* tmp = head;

while (tmp) {

tmp->item->get\_info();

tmp = tmp->next;

}

std::cout << '\n';

}

void moveable(int x, int y) {

std::cout << "Могут ли фигуры передвинуться на позицию (" << x << "," << y << "):\n";

Elem\* tmp = head;

while (tmp) {

if (tmp->item->moveable(x, y))

std::cout << "Можно! ";

else

std::cout << "Нельзя! ";

tmp->item->get\_info();

tmp = tmp->next;

}

}

};

int main() {

setlocale(LC\_ALL, "Russian");

Checker\* c1 = new Checker(false, 1, 6), \*c2 = new Checker(true, 2, 4), \*c3 = new Checker(false, 5, 2);

King \*k1 = new King(false, 1, 1), \*k2 = new King(true, 2, 7), \*k3 = new King(true, 9, 2);

Board\* board = new Board;

board->get\_info();

board->Add\_chess(c1);

board->Add\_chess(c3);

board->Add\_chess(k1, 6, 6);

board->Add\_chess(c2);

board->Add\_chess(k2);

board->Add\_chess(k3);

board->get\_info();

board->Delete\_chess();

board->moveable(2, 3);

}

