Московский государственный технический

университет им. Н.Э. Баумана.

Факультет «Информатика и системы управления»

Кафедра ИУ5. Курс «Программирование на основе классов и шаблонов»

Отчет по лабораторной работе № 1

«Разработка и использование класса “Планета”»

Выполнил:
студент группы ИУ5-23
Терентьев Владислав
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Козлов А. Д.
Подпись и дата:

Москва, 2019 г.

1. Постановка задачи

Разработать класс “Planet”, который хранит информацию о планете (имя, диаметр, количество спутников, наличие жизни). Реализовать интерфейс управления БД из объектов класса Planet, введенных пользователем, с функциями: вывод информации о планетах; изменение номера, имени, количества спутников, наличия жизни планеты; сортировка планет по имени, диаметру, количеству спутников, наличию жизни; добавление и удаление планеты.

2. Описание алгоритма

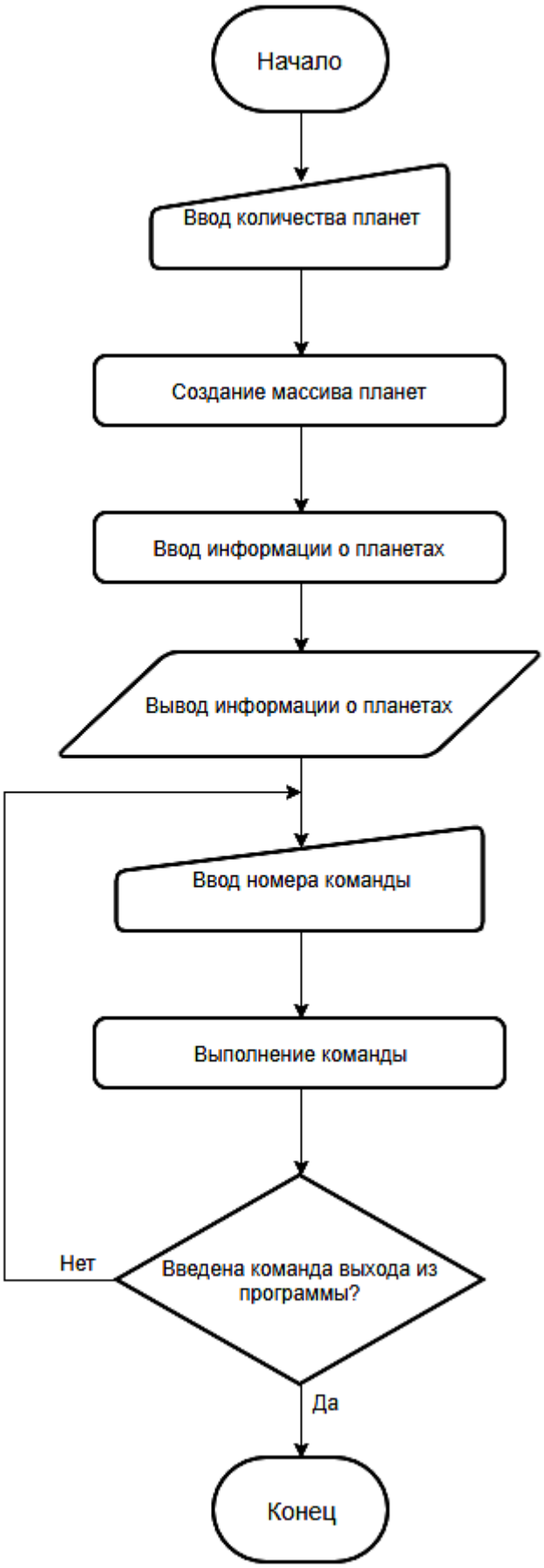
В программе описан класс **Planet** с переменными-членами (спецификатор доступа private): Number типа int (номер планеты), Name типа string (имя планеты), Diameter типа long int (диаметр планеты), Satellites типа int (количество спутников планеты), Life типа bool (наличие жизни на планете). Класс содержит **методы** (спецификатор доступа public): getters и setters для переменных-членов класса; inputName типа void и параметрами (указатель на массив Planet, количество планет) для ввода имени планеты; inputDiameter типа void для ввода диаметра планеты; inputSatellites типа void для ввода количества спутников планеты; inputLife типа void для ввода информации о наличии жизни на планете; inputInfoPlanet типа void и параметрами (указатель на массив Planet, количество планет) для ввода данных о планете; changeName типа void и параметрами (указатель на массив Planet, количество планет) для изменения имени планеты; infoPlanet типа void для вывода информации о планете.

В программе описаны **функции** типа void и параметрами (указатель на массив Planet, количество планет): sortDiameter для сортировки массива планет по диаметру; sortSatellites для сортировки массива планет по количеству спутников; sortName для сортировки массива планет по имени; sortLife для сортировки массива планет по наличию жизни. И **функция** searchPlanet типа int и параметрами (указатель на массив Planet, количество планет) для поиска планеты в массиве по имени.

В функции **main** программа просит пользователя ввести количество планет и заполнить информацию о них. После чего она выводит список команд, доступных пользователю для управления созданной БД планет.

```
1 - вывести информацию о всех планетах
2 - изменить номер планеты
3 - изменить имя планеты
4 - изменить диаметр планеты
5 - изменить количество спутников планеты
6 - изменить наличие жизни планеты
7 - сортировка планет по имени
8 - сортировка планет по диаметру
9 - сортировка планет по количеству спутников
10 - сортировка планет по наличию жизни
11 - добавить новую планету
12 - удалить планету
0 - выйти из программы.
```

Схема алгоритма:



3. Текст программы

```
#include <iostream>
#include <string>

using namespace std;

class Planet {
private:
	int Number;
	string Name;
	long int Diameter;
	int Satellites;
	bool Life;
public:
	int getNumber() {
		return Number;
	};
	string getName() {
		return Name;
	};
	long int getDiameter() {
		return Diameter;
	};
	int getSatellites() {
		return Satellites;
	};
	bool getLife() {
		return Life;
	};
	void setNumber(int newNumber) {
		Number = newNumber;
	};
	void setName(string newName) {
		Name = newName;
	};
	short setDiameter(long int newDiameter) {
		if (newDiameter < 1) {
			return 1;
		}
		else {
			Diameter = newDiameter;
			return 0;
		}
	};
	short setSatellites(int newSatellites) {
		if (newSatellites < 0) {
			return 1;
		}
		else {
			Satellites = newSatellites;
			return 0;
		}
	};
	short setLife(int newLife) {
		if (newLife != 0 && newLife != 1) {
			return 1;
		}
		else {
			Life = bool(newLife);
```

```
                return 0;
        }
};
void inputName(Planet* solarSystem, int num) {
        string newName, temp;
        int i;
        short res;
        num--;
        do {
                cout << "Введите имя планеты: ";
                cin >> newName;
                res = 0;
                for (i = 0; i < num; i++) {
                        temp = solarSystem[i].getName();
                        if (newName == temp) {
                                cout << "Имя планеты не должно совпадать с другими. ";
                                res = 1;
                                break;
                        }
                }
        } while (res != 0);
        setName(newName);
};
void inputDiameter() {
        int res;
        long int newDiameter;
        do {
                cout << "Введите диаметр планеты (диаметр должен быть больше 0): ";
                cin >> newDiameter;
                res = setDiameter(newDiameter);
                if (res != 0) {
                        cout << "Проверьте правильность данных (диаметр должен быть больше 0). ";
                }
        } while (res != 0);
};
void inputSatellites() {
        int res;
        int newSatellites;
        do {
                cout << "Введите количество спутников планеты (количество не должно быть
отрицательным): ";
                cin >> newSatellites;
                res = setSatellites(newSatellites);
                if (res != 0) {
                        cout << "Проверьте правильность данных (количество не должно быть
отрицательным). ";
                }
        } while (res != 0);
};
void inputLife() {
        int res, newLife;
        do {
                cout << "На планете есть жизнь? (1 - есть, 0 - нет): ";
                cin >> newLife;
                res = setLife(newLife);
                if (res != 0) {
                        cout << "Проверьте правильность данных. ";
                }
        } while (res != 0);
};
```

```
	void inputInfoPlanet(Planet * arr, int num) {
		cout << "Заполните информацию о планете номер " << num << ":" << endl;
		setNumber(num);
		inputName(arr, num);
		inputDiameter();
		inputSatellites();
		inputLife();
		cout << endl;
	};
	void changeName(Planet * solarSystem, int numPlanets) {
		string newName, temp;
		int i;
		short res;
		do {
			cout << "Введите новое имя планеты: ";
			cin >> newName;
			res = 0;
			for (i = 0; i < getNumber() - 1; i++) {
				temp = solarSystem[i].getName();
				if (newName == temp) {
					cout << "Имя планеты не должно совпадать с другими. ";
					res = 1;
					break;
				}
			}
			if (res == 0) {
				for (i++; i < numPlanets; i++) {
					temp = solarSystem[i].getName();
					if (newName == temp) {
						cout << "Имя планеты не должно совпадать с другими. ";
						res = 1;
						break;
					}
				}
			}
		} while (res != 0);
		setName(newName);
		cout << endl;
	}

	void infoPlanet() {
		cout << "Планета номер " << getNumber() << ":" << endl;
		cout << "Имя: " << getName() << endl;
		cout << "Диаметр: " << getDiameter() << endl;
		cout << "Количество спутников: " << getSatellites() << endl;
		cout << "Наличие жизни: " << boolalpha << getLife() << endl << endl;
	}

};

void sortDiameter(Planet*, int);
void sortSatellites(Planet*, int);
void sortName(Planet*, int);
void sortLife(Planet*, int);
int searchPlanet(Planet*, int);

void main()
{
	setlocale(LC_ALL, "Russian");
	int numPlanets, i, tar, j, curnum;
```

```
	string newName2, stemp;
	Planet* solarSystem = NULL;
	short res;
	cout << "Введите количество планет: ";
	cin >> numPlanets;
	cout << endl;
	solarSystem = new Planet[numPlanets];
	for (i = 0; i < numPlanets; i++) {
		solarSystem[i].inputInfoPlanet(solarSystem, i + 1);
	}
	for (i = 0; i < numPlanets; i++) {
		solarSystem[i].infoPlanet();
	}
	do {
		cout << endl << "Введите номер команды.\n1 - вывести информацию о всех планетах\n2 - изменить номер планеты\n3 - изменить имя планеты\n4 - изменить диаметр планеты\n5 - изменить количество спутников планеты\n6 - изменить наличие жизни планеты\n7 - сортировка планет по имени\n8 - сортировка планет по диаметру\n9 - сортировка планет по количеству спутников\n10 - сортировка планет по наличию жизни\n11 - добавить новую планету\n12 - удалить планету\n0 - выйти из программы." << endl;
		cin >> tar;
		cout << endl;
		switch (tar) {
		case 1: {
			for (i = 0; i < numPlanets; i++) {
				solarSystem[i].infoPlanet();
			}
			break;
		}
		case 2: {
			i = searchPlanet(solarSystem, numPlanets);
			do {
				res = 0;
				cout << "Введите новый номер этой планеты (номер планеты должен быть в пределе от 1 до " << numPlanets << "): " << endl;
				cin >> curnum;
				if (curnum > 0 && curnum <= numPlanets) {
					i--;
					solarSystem[i].setNumber(curnum);
					curnum--;
					if (i < curnum) {
						for (i; i < curnum; i++) {
							swap(solarSystem[i], solarSystem[i + 1]);
							solarSystem[i].setNumber(i + 1);
						}
					}
					else {
						for (i; i > curnum; i--) {
							swap(solarSystem[i], solarSystem[i - 1]);
							solarSystem[i].setNumber(i + 1);
						}
					}
					res = 1;
				}
				else {
					cout << "Проверьте правильность данных (номер планеты должен быть в пределе от 1 до " << numPlanets << "). Попробуйте заново." << endl;
				}
			} while (res != 1);
			cout << "Порядок планет изменен." << endl;
```

```
			break;
		}
		case 3: {
			i = searchPlanet(solarSystem, numPlanets);
			solarSystem[i - 1].changeName(solarSystem, numPlanets);
			cout << "Имя планеты изменено." << endl;
			break;
		}
		case 4: {
			i = searchPlanet(solarSystem, numPlanets);
			solarSystem[i - 1].inputDiameter();
			cout << "Диаметр планеты изменен." << endl;
			break;
		}
		case 5: {
			i = searchPlanet(solarSystem, numPlanets);
			solarSystem[i - 1].inputSatellites();
			cout << "Количество спутников планеты изменено." << endl;
			break;
		}
		case 6: {
			i = searchPlanet(solarSystem, numPlanets);
			solarSystem[i - 1].inputLife();
			cout << "Наличие жизни на планете изменено." << endl;
			break;
		}
		case 7: {
			sortName(solarSystem, numPlanets);
			break;
		}
		case 8: {
			sortDiameter(solarSystem, numPlanets);
			break;
		}
		case 9: {
			sortSatellites(solarSystem, numPlanets);
			break;
		}
		case 10: {
			sortLife(solarSystem, numPlanets);
			break;
		}
		case 11: {
			do {
				cout << "Введите номер новой планеты (всего планет " << numPlanets << "):
" << endl;
				cin >> i;
				Planet* temp = new Planet[numPlanets + 1];
				if (i > 0 && i <= numPlanets + 1) {
					numPlanets++;
					for (j = 0; j < i - 1; j++) {
						temp[j] = solarSystem[j];
					}
					for (j = i - 1; j < numPlanets - 1; j++) {
						solarSystem[j].setNumber(j + 2);
						temp[j + 1] = solarSystem[j];
					}
					cout << "Заполните информацию о новой планете: " << endl;
					temp[i - 1].setNumber(i);
					do {
```

```
                                        cout << "Введите имя планеты: ";
                                        cin >> newName2;
                                        res = 0;
                                        for (j = 0; j < numPlanets - 1; j++) {
                                               stemp = solarSystem[j].getName();
                                               if (newName2 == stemp) {
                                                      cout << "Имя планеты не должно совпадать с
другими. ";
                                                      res = 1;
                                                      break;
                                               }
                                        }
                                 } while (res != 0);
                                 temp[i - 1].setName(newName2);
                                 temp[i - 1].inputDiameter();
                                 temp[i - 1].inputSatellites();
                                 temp[i - 1].inputLife();
                                 cout << endl;
                                 delete[]solarSystem;
                                 solarSystem = temp;
                                 temp = NULL;
                                 cout << "Новая планета создана." << endl;
                          }
                          else {
                                 cout << "Проверьте правильность данных. ";
                          }
                   } while (i <= 0 || i > numPlanets + 1);
                   break;
            }
            case 12: {
                   i = searchPlanet(solarSystem, numPlanets);
                   for (j = i - 1; j < numPlanets - 1; j++) {
                          swap(solarSystem[j], solarSystem[j + 1]);
                          solarSystem[j].setNumber(j + 1);
                   }
                   numPlanets--;
                   cout << "Планета удалена." << endl;
                   break;
            }

            case 0: break;

            default: {
                   cout << "Проверьте правильность данных." << endl;
            }

            }
      } while (tar != 0);
      delete[] solarSystem;
}

void sortDiameter(Planet * arr, int nm) {
      int ij;
      unsigned int* temp = new unsigned int[nm];
      for (int k = 0; k < nm; k++) {
            temp[k] = k;
      }
      cout << "Сортировка по диаметру" << endl;
      for (int i = 0; i < nm - 1; i++) {
            ij = i;
```

```
		for (int j = i + 1; j < nm; j++) {
			if (arr[temp[j]].getDiameter() > arr[temp[ij]].getDiameter()) {
				ij = j;
			}
			else {
				if (arr[temp[j]].getDiameter() == arr[temp[ij]].getDiameter()) {
					if (arr[temp[j]].getNumber() < arr[temp[ij]].getNumber()) {
						ij = j;
					}
				}
			}
		}
		if (ij != i) {
			swap(temp[i], temp[ij]);
		}
		arr[temp[i]].infoPlanet();
	}
	arr[temp[nm - 1]].infoPlanet();
	delete[]temp;
	temp = NULL;
}

void sortSatellites(Planet * arr, int nm) {
	int ij;
	unsigned int* temp = new unsigned int[nm];
	for (int k = 0; k < nm; k++) {
		temp[k] = k;
	}
	cout << "Сортировка по спутникам" << endl;
	for (int i = 0; i < nm - 1; i++) {
		ij = i;
		for (int j = i + 1; j < nm; j++) {
			if (arr[temp[j]].getSatellites() > arr[temp[ij]].getSatellites()) {
				ij = j;
			}
			else {
				if (arr[temp[j]].getSatellites() == arr[temp[ij]].getSatellites()) {
					if (arr[temp[j]].getNumber() < arr[temp[ij]].getNumber()) {
						ij = j;
					}
				}
			}
		}
		if (ij != i) {
			swap(temp[i], temp[ij]);
		}
		arr[temp[i]].infoPlanet();
	}
	arr[temp[nm - 1]].infoPlanet();
	delete[]temp;
	temp = NULL;
}

void sortName(Planet * arr, int nm) {
	short res;
	int ij;
	unsigned int* temp = new unsigned int[nm];
	for (int k = 0; k < nm; k++) {
		temp[k] = k;
	}
```

```
	cout << "Сортировка по имени" << endl;
	for (int i = 0; i < nm - 1; i++) {
		ij = i;
		for (int j = i + 1; j < nm; j++) {
			res = strcoll(arr[temp[j]].getName().c_str(), arr[temp[ij]].getName().c_str());
			if (res == -1) {
				ij = j;
			}
		}
		if (ij != i) {
			swap(temp[i], temp[ij]);
		}
		arr[temp[i]].infoPlanet();
	}
	arr[temp[nm - 1]].infoPlanet();
	delete[]temp;
	temp = NULL;
}

void sortLife(Planet * arr, int nm) {
	int ij;
	unsigned int* temp = new unsigned int[nm];
	for (int k = 0; k < nm; k++) {
		temp[k] = k;
	}
	cout << "Сортировка по наличию жизни" << endl;
	for (int i = 0; i < nm - 1; i++) {
		ij = i;
		for (int j = i + 1; j < nm; j++) {
			if (arr[temp[j]].getLife() > arr[temp[ij]].getLife()) {
				ij = j;
			}
			else {
				if (arr[temp[j]].getLife() == arr[temp[ij]].getLife()) {
					if (arr[temp[j]].getNumber() < arr[temp[ij]].getNumber()) {
						ij = j;
					}
				}
			}
		}
		if (ij != i) {
			swap(temp[i], temp[ij]);
		}
		arr[temp[i]].infoPlanet();
	}
	arr[temp[nm - 1]].infoPlanet();
	delete[]temp;
	temp = NULL;
}

int searchPlanet(Planet * arr, int numPlanets) {
	string Name;
	short res = 0;
	int i;
	do {
		cout << "Введите имя нужной планеты: ";
		cin >> Name;
		for (i = 0; i < numPlanets; i++) {
			if (arr[i].getName() == Name) {
				res = 1;
```

```
                        break;
                }
            }
            if (res != 1) {
                cout << "Планета с таким именем не найдена. Попробуйте заново." << endl;
            }
        } while (res != 1);
        return i + 1;
}
```

4. Анализ результатов

Проверка ввода допустимых значений:

```
C:\Users\webma\source\repos\lab1\Debug\lab1.exe
Введите количество планет: 3

Заполните информацию о планете номер 1:
Введите имя планеты: Mercury
Введите диаметр планеты (диаметр должен быть больше 0): 0
Проверьте правильность данных. Введите диаметр планеты (диаметр должен быть больше 0): 4879
Введите количество спутников планеты (количество не должно быть отрицательным): -2
Проверьте правильность данных. Введите количество спутников планеты (количество не должно быть отрицательным): 0
На планете есть жизнь? (1 - есть, 0 - нет): 3
Проверьте правильность данных. На планете есть жизнь? (1 - есть, 0 - нет): 0

Заполните информацию о планете номер 2:
Введите имя планеты: Mercury
Имя планеты не должно совпадать с другими. Введите имя планеты: Venus
Введите диаметр планеты (диаметр должен быть больше 0): 12103
Введите количество спутников планеты (количество не должно быть отрицательным): 0
На планете есть жизнь? (1 - есть, 0 - нет): 0

Заполните информацию о планете номер 3:
Введите имя планеты: Earth
Введите диаметр планеты (диаметр должен быть больше 0): 12742
Введите количество спутников планеты (количество не должно быть отрицательным): 1
На планете есть жизнь? (1 - есть, 0 - нет): 1
```

Проверка работы команд:

```
3

Введите имя нужной планеты: Earth
Введите новое имя планеты: Venus
Имя планеты не должно совпадать с другими. Введите новое имя планеты: Earth1

Имя планеты изменено.
```

```
Планета номер 3:
Имя: Earth1
Диаметр: 12742
Количество спутников: 1
Наличие жизни: true
```

```
9

Сортировка по спутникам
Планета номер 3:
Имя: Earth1
Диаметр: 12742
Количество спутников: 1
Наличие жизни: true

Планета номер 1:
Имя: Mercury
Диаметр: 4879
Количество спутников: 0
Наличие жизни: false

Планета номер 2:
Имя: Venus
Диаметр: 12103
Количество спутников: 0
Наличие жизни: false
```

```
11

Введите номер новой планеты (всего планет 3):
2
Заполните информацию о новой планете:
Введите имя планеты: Venus
Имя планеты не должно совпадать с другими. Введите имя планеты: Mars
Введите диаметр планеты (диаметр должен быть больше 0): 6780
Введите количество спутников планеты (количество не должно быть отрицательным): 2
На планете есть жизнь? (1 - есть, 0 - нет): 1

Новая планета создана.
```

```
Планета номер 2:
Имя: Mars
Диаметр: 6780
Количество спутников: 2
Наличие жизни: true
```

```
12

Введите имя нужной планеты: Earth
Планета с таким именем не найдена. Попробуйте заново.
Введите имя нужной планеты: Earth1
Планета удалена.
```

```
1

Планета номер 1:
Имя: Mercury
Диаметр: 4879
Количество спутников: 0
Наличие жизни: false

Планета номер 2:
Имя: Mars
Диаметр: 6780
Количество спутников: 2
Наличие жизни: true

Планета номер 3:
Имя: Venus
Диаметр: 12103
Количество спутников: 0
Наличие жизни: false
```

Программа работает исправно.