3. Семинар: классы и объекты

3.1. Инструментарий

— Презентация для преподавателя, ведущего семинар;
— Фон GeekBrains для проведения семинара в Zoom;
— JDK любая 11 версии и выше;
— IntelliJ IDEA Community Edition для практики и примеров используется IDEA.

3.2. Цели семинара

— Закрепить полученные на лекции знания об объектах, наследовании и полиморфизме;
— Получить практический навык создания классов по описанию;
— Дополнительно рассмотреть использование свойств статичности сущностей, неочевидные случаи несоблюдения инкапсуляции;
— Попрактиковаться в написании простых классов и методов, манипулирующих ссылочными данными.

3.3. План-содержание

Что происходит	Время	Слайды	Описание
Организационный момент	5	1-5	Преподаватель ожидает студентов, поддерживает активность и коммуникацию в чате, озвучиает цели и планы на семинар. Важно упомянуть, что выполнение домашних заданий с лекции является, фактически, подготовкой к семинару
Quiz	10	6-24	Преподаватель задаёт вопросы викторины, через 30 секунд демонстрирует слайд-подсказку и ожидает ответов (6 вопросов, по минуте на ответ)
Рассмотрение ДЗ лекции	10	25-30	Преподаватель демонстрирует свой вариант решения домашнего задания с лекции, возможно, по предварительному опросу, демонстрирует и разбирает вариант решения одного из студентов
Вопросы и ответы	10	31	Преподаватель ожидает вопросов по теме прошедшей лекции, викторины и продемонстрированной работы
Задание 1	5	32-35	Создание класса и объекта.

Что происходит	Время	Слайды	Описание
Задание 2	10	36-40	Манипуляция информацией об объекте
Перерыв (если нужен)	5	41	Преподаватель предлагает студентам перерыв на 5 минут (студенты голосуют)
Задание 3	20	42-46	Создание и манипуляция множествами объектов
Задание 4	15	47-51	Манипуляции данными по условию, «массовое обслуживание»
Задание 5 (необязат)	15	52-54	Соблюдение атомарности методов, независимость методов от окружения
Домашнее задание	5	55-56	Объясните домашнее задание, подведите итоги урока
Рефлексия	10	57-58	Преподаватель запрашивает обратную связь
Длительность	120		

3.4. Подробности

Организационный момент

- **Цель этапа:** Позитивно начать урок, создать комфортную среду для обучения.
- **Тайминг:** 3-5 минут.
- **Действия преподавателя:**
  - Запрашивает активность от аудитории в чате;
  - Презентует цели курса и семинара;
  - Презентует краткий план семинара и что студент научится делать.

Quiz

- **Цель этапа:** Вовлечение аудитории в обратную связь.
- **Тайминг:** 5-7 минут (4 вопроса, по минуте на ответ).
- **Действия преподавателя:**
  - Преподаватель задаёт вопросы викторины, представленные на слайдах презентации;
  - через 30 секунд демонстрирует слайд-подсказку и ожидает ответов.
- **Вопросы и ответы:**
  1. Какое свойство добавляет ключевое слово static полю или методу? (2)
     (a) неизменяемость;
     (b) принадлежность классу;
     (c) принадлежность приложению.
  2. Что быстрее, стек или куча? (1)
     (a) куча;
     (b) стек;
     (c) одинаково.

3. Для инициализации нового объекта абсолютно идентичными значениями свойств переданного объекта используется (3)
   (a) пустой конструктор;
   (b) конструктор по-умолчанию;
   (c) конструктор копирования.
4. Инкапсуляция – это (2)
   (a) архивирование проекта
   (b) сокрытие информации о классе
   (c) создание микросервисной архитектуры
5. Наследуются от Object (3)
   (a) строки
   (b) потоки ввода-вывода
   (c) и то, и другое
6. Является ли перегрузка полиморфизмом (2)
   (a) да, это истинный полиморфизм
   (b) да, это часть истинного полиморфизма
   (c) нет, это не полиморфизм

Рассмотрение ДЗ

— **Цель этапа:** Пояснить не очевидные моменты в формулировке ДЗ с лекции, синхронизировать прочитанный на лекции материал к началу семинара.
— **Тайминг:** 15-20 минут.
— **Действия преподавателя:**
  – Преподаватель демонстрирует свой вариант решения домашнего задания из лекции;
  – возможно, по предварительному опросу, демонстрирует и разбирает вариант решения одного из студентов.
— **Домашнее задание из лекции:**
  – Написать класс кота так, чтобы каждому объекту кота присваивался личный порядковый целочисленный номер;
    Вариант решения в приложении А
  – Написать классы кота и собаки, наследники животного. У всех есть три действия: бежать, плыть, прыгать. Действия принимают размер препятствия и возвращают булев результат. Три ограничения: высота прыжка, расстояние, которое животное может пробежать, расстояние, которое животное может проплыть. Следует учесть, что коты не любят воду.
    Вариант решения в приложении Б
  – Добавить механизм, создающий 25% разброс значений каждого ограничения для каждого объекта.
    Вариант решения

Листинг 1: Конструктор с вариативностью

```
    Animal(String type, String name, float maxJump, float maxRun, float maxSwim) {
        float jumpDiff = random.nextFloat() * maxJump - (maxJump / 2);
        float runDiff = random.nextFloat() * maxRun - (maxRun / 2);
        float swimDiff = random.nextFloat() * maxSwim - (maxSwim / 2);

        this.type = type;
        this.name = name;
        this.maxJump = maxJump + jumpDiff;
        this.maxRun = maxRun + runDiff;
        this.maxSwim = maxSwim + swimDiff;
    }
```

Вопросы и ответы

- **Ценность этапа** Вовлечение аудитории в обратную связь, пояснение неочевидных моментов в материале лекции и другой проделанной работе.
- **Тайминг** 5-15 минут
- **Действия преподавателя**
  - Преподаватель ожидает вопросов по теме прошедшей лекции, викторины и продемонстрированной работы;
  - Если преподаватель затрудняется с ответом, необходимо мягко предложить студенту ответить на его вопрос на следующем семинаре (и не забыть найти ответ на вопрос студента!);
  - Предложить и показать пути самостоятельного поиска студентом ответа на заданный вопрос;
  - Посоветовать литературу на тему заданного вопроса;
  - Дополнительно указать на то, что все сведения для выполнения домашнего задания, прохождения викторины и работы на семинаре были рассмотрены в методическом материале к этому или предыдущим урокам.

Задание 1

- **Ценность этапа** Создание класса и объекта.
- **Тайминг** 5 минут.
- **Действия преподавателя**
  - Первые пять минут уклоняться от ответов на уточняющие вопросы
  - Выдать задание студентам;
- **Задания**:
  - Создать класс ”Сотрудник”с полями: ФИО, должность, телефон, зарплата, возраст;
    Вариант исполнения класса в приложении В

Листинг 2: Создание объекта класса

```
Employee employeeIvan = new Employee("Ivan", "Igorevich",
      "Ovchinnikov", "8(495)000-11-22",
      "developer", 50000, 1985);
```

Задание 2

- **Ценность этапа** Манипуляция информацией об объекте.
- **Тайминг** 10 минут.
- **Действия преподавателя**
  - Пояснить студентам ценность этого опыта (отказ от вывода информации в терминал из сторонних объектов);
  - Пояснить студентам в каком виде выполнять и сдавать задания;
  - Выдать задание группам студентов;
  - Если группа студентов справилась с заданием, а времени осталось более 5 минут, выдавать группе задания «со звёздочкой».
- **Задания**
  - Написать функцию выводящую всю доступную информацию об объекте
    Вариант решения

Листинг 3: Вывод информации об объекте в консоль

```
   public void info() {
      System.out.println("Employee{" +
            "name='" + name + '\'' +
            ", midName='" + midName + '\'' +
            ", surName='" + surName + '\'' +
            ", position='" + position + '\'' +
            ", phone='" + phone + '\'' +
            ", salary=" + salary +
            ", age=" + getAge() +
            '}');
   }
```

$*_1$ таким образом, чтобы функция возвращала значение;
Вариант решения

Листинг 4: Возврат информации об объекте

```
   @Override
   public String toString() {
      return "Employee{" +
            "name='" + name + '\'' +
            ", midName='" + midName + '\'' +
            ", surName='" + surName + '\'' +
            ", position='" + position + '\'' +
            ", phone='" + phone + '\'' +
            ", salary=" + salary +
            ", age=" + getAge() +
            '}';
   }
```

$*_2$ с использованием форматирования строк.
Вариант решения

Листинг 5: Форматированная информация об объекте

```
   @Override
   public String toString() {
      return String.format("Employee{" +
            "name='%s', midName='%s', surName='%s'" +
```

```
            ", position='%s', phone='%s'" +
            ", salary=%d, age=%d}'",
            name, midName, surName, position, phone, salary, getAge());
    }
```

Задание 3

- **Ценность этапа** Создание и манипуляция множествами объектов.
- **Тайминг** 20-25 минут.
- **Действия преподавателя**
  - Пояснить студентам ценность этого опыта (дополнительно напомнить, что классы создают для программы новый тип данных, а значит объектами можно пользоваться также, как и заранее созданными);
  - Пояснить студентам в каком виде выполнять и сдавать задания;
  - Выдать задание группам студентов;
  - Если группа студентов справилась с заданием, а времени осталось более 5 минут, выдавать группе задания «со звёздочкой».
- **Задания**
  - Создать массив из 5 сотрудников
    Вариант решения

Листинг 6: Простой массив сотрудников

```
Employee ivan = new Employee("Ivan", "Igorevich",
      "Ovchinnikov", "8(495)000-11-22",
      "developer", 50000, 1985);
Employee andrey = new Employee("Andrey", "Viktorovich",
      "Bezrukov", "8(495)111-22-33",
      "fitter", 52000, 1973);
Employee evgeniy = new Employee("Evgeniy", "Viktorovich",
      "Delfinov", "8(495)222-33-44",
      "project manager", 40000, 1963);
Employee natalia = new Employee("Natalia", "Pavlovna",
      "Keks", "8(495)333-44-55",
      "senior developer", 90000, 1990);
Employee tatiana = new Employee("Tatiana", "Sergeevna",
      "Krasotkina", "8(495)444-55-66",
      "accountant", 50000, 1983);

Employee[] company = new Employee[5];
company[0] = ivan;
company[1] = andrey;
company[2] = evgeniy;
company[3] = natalia;
company[4] = tatiana;
```

  $*_1$ массив должен быть сразу инициализирован;
  Вариант решения

Листинг 7: Инициализированный массив

```
Employee[] employees = { ivan, andrey, evgeniy, natalia, tatiana };
```

$*_2$ массив должен быть сразу инициализирован и не должно быть создано дополнительных переменных.

Вариант решения

Листинг 8: Без дополнительных переменных

```
Employee[] employees = {
      new Employee("Ivan", "Igorevich",
            "Ovchinnikov", "8(495)000-11-22",
            "developer", 50000, 1985),
      new Employee("Andrey", "Viktorovich",
            "Bezrukov", "8(495)111-22-33",
            "fitter", 52000, 1973),
      new Employee("Evgeniy", "Viktorovich",
            "Delfinov", "8(495)222-33-44",
            "project manager", 40000, 1963),
      new Employee("Natalia", "Pavlovna",
            "Keks", "8(495)333-44-55",
            "senior developer", 90000, 1990),
      new Employee("Tatiana", "Sergeevna",
            "Krasotkina", "8(495)444-55-66",
            "accountant", 50000, 1983)
 };
```

Задание 4

- **Ценность этапа** Манипуляция информацией об объекте.
- **Тайминг** 15-20 минут.
- **Действия преподавателя**
  - Пояснить студентам ценность этого опыта (часто возникают ситуации, когда необходимо осуществить фильтрацию данных или манипулировать только частью объектов. дополнительно указать на то, что часто методы, манипулирующие множеством объектов описывают внутри классов, но это не совсем верный подход с точки зрения архитектуры, и под такие методы желательно создавать отдельный класс, например, `EmployeeWorker`);
  - Пояснить студентам в каком виде выполнять и сдавать задания;
  - Выдать задание группам студентов;
  - Если группа студентов справилась с заданием, а времени осталось более 5 минут, выдавать группе задания «со звёздочкой».
- **Задания**
  - Создать метод, повышающий зарплату всем сотрудникам старше 45 лет на 5000. Метод должен принимать в качестве параметра массив сотрудников.

    Вариант решения

Листинг 9: Метод повышения зарплаты

```
// Employee
public void increaseSalary(int amount) {
   this.salary += amount;
}
// Main
private static void increaser(Employee[] emp) {
```

```
   for (int i = 0; i < emp.length; i++) {
      if (emp[i].getAge() > 45) {
         emp[i].increaseSalary(5000);
      }
   }
}
// main
for (int i = 0; i < employees.length; i++) {
   increaser(employees);
}
```

$*_1$ Написать тот же метод, но возраст и размер повышения должны быть параметрами метода.

Вариант решения

Листинг 10: Параметризированный метод повышения

```
// Main
private static void increaser(Employee[] emp, int age, int increment) {
   for (int i = 0; i < emp.length; i++) {
      if (emp[i].getAge() > age) {
         emp[i].increaseSalary(increment);
      }
   }
}
// main
for (int i = 0; i < employees.length; i++) {
   increaser(employees, 45, 5000);
}
```

$*_2$ Написать тот же метод в качестве статического в классе сотрудника.

Вариант решения

Листинг 11: Статический метод и использование

```
// Employee
public static void increaser(Employee[] emp, int age, int increment) {
   for (int i = 0; i < emp.length; i++) {
      if (emp[i].getAge() > age) {
         emp[i].increaseSalary(increment);
      }
   }
}
// main
for (int i = 0; i < employees.length; i++) {
   Employee.increaser(employees, 45, 5000);
}
```

Задание 5

- **Ценность этапа** Соблюдение атомарности методов, независимость методов, манипулирующих данными от окружения.
- **Тайминг** 10-15 минут.
- **Действия преподавателя**
  - Обратить внимание на то, что многие студенты пытаются объединить методы с похожими алгоритмами, но это неверный подход, нужно соблюдать

атомарность методов. Важно, чтобы результаты работы не печатались в консоль, а возвращались из методов, чтобы код был переносимым;
  - Выдать задание группам студентов;
  - Если группа студентов справилась с заданием, а времени осталось более 5 минут, выдавать группе задания «со звёздочкой».
- **Задания**
  - Написать методы (принимающие на вход массив сотрудников), вычисляющие средний возраст и среднюю зарплату сотрудников, вывести результаты работы в консоль.

    Вариант решения

Листинг 12: Методы подсчёта средних

```
private static float averageSalary(Employee[] emp) {
   float result = 0;
   for (int i = 0; i < emp.length; i++)
      result += emp[i].getSalary();

   return result / emp.length;
}

private static float averageAge(Employee[] emp){
   float result = 0;
   for (int i = 0; i < emp.length; i++)
      result += emp[i].getAge();

   return result / emp.length;
}
```

Домашнее задание

- **Ценность этапа** Задать задание для самостоятельного выполнения между занятиями.
- **Тайминг** 5-10 минут.
- **Действия преподавателя**
  * Пояснить студентам в каком виде выполнять и сдавать задания
  * Уточнить кто будет проверять работы (преподаватель или ревьювер)
  * Объяснить к кому обращаться за помощью и где искать подсказки
  * Объяснить где взять проект заготовки для дз
- **Задания**

5-25 мин Решить все задания (в том числе «со звёздочкой»), если они не были решены на семинаре, без ограничений по времени;

Все варианты решения приведены в тексте семинара выше

5-10 мин 1. Написать прототип компаратора - метод внутри класса сотрудника, сравнивающий две даты, представленные в виде трёх чисел гггг-мм-дд, без использования условного оператора.

Листинг 13: Сравнение возраста

```
//Employee
```

```
int bMonth;
int bDay;
/**
 * returns
 * negative integer if this object birthdate is less (earlier), than given (older)
 * positive integer if this object birthdate is more (later), than given (younger)
 * zero if this object is the same as given
 * */
public int compare(int dd, int mm, int yyyy) {
   //day = 0..30, 31 is binary 11111, ok to left shift month by 6
   //month = 0..11, 15 is binary 1111, ok to left shift year by 5 more
   int empl = bDay + (bMonth << 6) + (birth << 11);
   int income = dd + (mm << 6) + (yyyy << 11);
   return empl - income;
}
```

10-15 мин 2. Опишите класс руководителя, наследник от сотрудника. Перенесите статический метод повышения зарплаты в класс руководителя, модифицируйте метод таким образом, чтобы он мог поднять заработную плату всем, кроме руководителей. В основной программе создайте руководителя и поместите его в общий массив сотрудников. Повысьте зарплату всем сотрудникам и проследите, чтобы зарплата руководителя не повысилась.

Листинг 14: Менеджер - это тоже сотрудник

```
// Manager
package ru.gb.jcore;

public class Manager extends Employee {

   public Manager(String name, String midName, String surName,
               String phone, String position, int salary, int birth) {
      super(name, midName, surName, phone, position, salary, birth);
   }
   public static void increaser(Employee[] emp, int age, int increment) {
      for (int i = 0; i < emp.length; i++) {
         if (emp[i].getAge() > age) {
            if (!(emp[i] instanceof Manager))
               emp[i].increaseSalary(increment);
         }
      }
   }
}
```

Листинг 15: Более верное повышение зарплаты

```
// main
Employee[] employees = { ivan, andrey, evgeniy /*new Manager(...)*/, natalia,
    tatiana };
for (int i = 0; i < employees.length; i++) {
   Manager.increaser(employees, 45, 5000);
}
```

Рефлексия и завершение семинара

- **Цель этапа:** Привести урок к логическому завершению, посмотреть что студентам удалось, что было сложно и над чем нужно еще поработать
- **Тайминг:** 5-10 минут
- **Действия преподавателя:**
  - Запросить обратную связь от студентов.
  - Подчеркните то, чему студенты научились на занятии.
  - Дайте рекомендации по решению заданий, если в этом есть необходимость
  - Дайте краткую обратную связь студентам.
  - Поделитесь ощущением от семинара.
  - Поблагодарите за проделанную работу.

Приложения

А. Домашнее задание 1

Листинг 16: Кот

```
package ru.gb.jcore;

public class Cat {
   private static final int CURRENT_YEAR = 2022;
   private static int id = 0;

   private String name;
   private String color;
   private int birthYear;
   public int uid;

   Cat (String name, String color, int age) {
      setBirth(age);
      this.name = name;
      this.color = color;
      this.uid = ++id;
   }
   private void setBirth(int age) {
      this.birthYear = CURRENT_YEAR - age;
   }
   public int getUid() {
      return uid;
   }
   public String getColor() {
      return color;
   }
   public int getAge() {
      return CURRENT_YEAR - birthYear;
   }
   public String getName() {
      return name;
   }
}
```

Б. Домашнее задание 2

Листинг 17: Общий класс животного

```
package ru.gb.jcore.marathon;

import java.util.Random;

public abstract class Animal {

   static final int SWIM_FAIL = 0;
   static final int SWIM_OK = 1;
   static final int SWIM_WTF = -1;

   private String type;
   private String name;
   private float maxRun;
   private float maxSwim;
   private float maxJump;
   private final Random random = new Random();
```

```

    Animal(String type, String name, float maxJump, float maxRun, float maxSwim) {
        this.type = type;
        this.name = name;
        this.maxJump = maxJump;
        this.maxRun = maxRun;
        this.maxSwim = maxSwim;
    }

    String getName() {
        return this.name;
    }
    String getType() {
        return this.type;
    }
    float getMaxRun() {
        return this.maxRun;
    }
    float getMaxSwim() {
        return this.maxSwim;
    }
    float getMaxJump() {
        return this.maxJump;
    }
    protected boolean run(float distance) {
        return (distance <= maxRun);
    }
    protected int swim(float distance) {
        return (distance <= maxSwim) ? SWIM_OK : SWIM_FAIL;
    }
    protected boolean jump(float height) {
        return (height <= maxJump);
    }
}
```

Листинг 18: Кот

```
package ru.gb.jcore.marathon;

public class Cat extends Animal {

    Cat(String name) {
        super("Cat", name, 2, 200, 1);
    }

    @Override
    protected int swim(float distance) {
        return Animal.SWIM_WTF;
    }
}
```

Листинг 19: Собака

```
package ru.gb.jcore.marathon;

class Dog extends Animal {

    Dog(String name, float maxJump, float maxRun, float maxSwim) {
        super("Dog", name, maxJump, maxRun, maxSwim);
    }

}
```

Листинг 20: Марафон

```
package ru.gb.jcore.marathon;

public class Marathon {
   public static void main(String[] args) {

      Cat c = new Cat("Barseek");
      Cat c0 = new Cat("Moorzeek");
      Dog d = new Dog("Toozeek", 0.5f, 500, 10);
      Dog d0 = new Dog("Shaareek", 0.5f, 500, 10);

      Animal[] arr = {c, c0, d, d0};
      float toJump = 1.5f;
      float toRun = 350;
      float toSwim = 5;

      for (int i = 0; i < arr.length; i++) {
         String nameString = arr[i].getType() + " " + arr[i].getName() + " can ";

         String eventName = String.format("jump max %.2fm. Tries to jump ", arr[i].getMaxJump());
         String eventResult = (arr[i].jump(toJump)) ? "succeed" : "fails";
         System.out.println(nameString + eventName + toJump + "m and " + eventResult);

         eventName = String.format("run max %.2fm. Tries to run ", arr[i].getMaxRun());
         eventResult = arr[i].run(toRun) ? "succeed" : "fails";
         System.out.println(nameString + eventName + toRun + "m and " + eventResult);

         int swimResult = arr[i].swim(toSwim);
         eventName = String.format("swim max %.2fm. Tries to swim ", arr[i].getMaxSwim());
         eventResult = (swimResult == Animal.SWIM_OK) ? "succeed" : "fails";
         if (swimResult == Animal.SWIM_WTF)
            eventResult = "too scared to enter the water";
         System.out.println(nameString + eventName + toSwim + "m and " + eventResult);
      }
   }
}
```

B. Практическое задание 1

Листинг 21: Сотрудник

```
package ru.gb.jcore;

public class Employee {
   private static final int CURRENT_YEAR = 2022;
   String name;
   String midName;
   String surName;
   String position;
   String phone;
   int salary;
   int birth;

   public Employee(String name, String midName, String surName,
                String phone, String position, int salary, int birth) {
      this.name = name;
      this.midName = midName;
      this.surName = surName;
      this.position = position;
      this.phone = phone;
      this.salary = salary;
```

```
        this.birth = birth;
    }

    public String getName() {
        return name;
    }

    public String getMidName() {
        return midName;
    }

    public String getSurName() {
        return surName;
    }

    public String getPosition() {
        return position;
    }

    public void setPosition(String position) {
        this.position = position;
    }

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public int getSalary() {
        return salary;
    }

    public void setSalary(int salary) {
        this.salary = salary;
    }

    public int getAge() {
        return CURRENT_YEAR - birth;
    }

}
```