УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Кафедра інформаційних технологій та програмної інженерії

КУРСОВА РОБОТА

з Об’єктно-орієнтованого програмування

на тему: Класовий тип для роботи з структурами типу «Бітовий вектор»

Студента II курсу _____ групи

спеціальності 121 «Інженерія програмного забезпечення»

Куриш В.В.

(прізвище та ініціали)

Керівник _____к.т.н., Пашкевич О.П.

Національна шкала _____

Кількість балів _____ Оцінка: ECTS _____

м. Івано-Франківськ

2021

Університет Короля Данила

назва вищого навчального закладу

Кафедра інформаційних технологій та програмної інженерії

Дисципліна Об’єктоно-орієнтоване програмування

Спеціальність 121 «Інженерія програмного забезпечення»

Курс II Група ІПЗс-2019 Семестр 4

ЗАВДАННЯ

на курсовий проект (роботу) студента

Куриш Володимир Володимирович

Прізвище, ім’я, по-батькові

1. Тема проекту (роботи) Класовий тип для роботи з структурами типу «Бітовий вектор»

2. Строк здачі студентом закінченого проекту (роботи) _____

3. Вихідні дані до проекту (роботи) _____

4. Зміст роботи (перелік питань по розділах, які потрібно розробити) _____

5. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсового проекту (роботи)	Строк виконання	Примітки
1	Ознайомлення та пошук інформації	16.05 17.05	
2	Розробка архітектури	17.05 18.05	
3	Написання основного класу	18.05 19.05	
4	Написання першої поливини методів	19.05 20.05	
5	Написання другої поливини методів	20.05 21.05	
6	Оформлення курсової роботи	21.05 22.05	
7	Оформлення курсової роботи	22.05 23.05	

Студент _____ _____
Підпис Прізвище, ініціали

Керівник _____ _____
Підпис Прізвище, ініціали

«____» _____ 202___ р.

ЗМІСТ

ВСТУП

Мова програмування Java підтримує великий набір операцій. Операції використовуються у виразах.

Всі операції діляться на 5 груп:

арифметичні операції (+ , - , * , / і інші);

порозрядні логічні операції (~ , & , | та інші);

логічні операції (& , | , && , || та інші);

операції відносини (== , ! = , > , < і інші);

додаткові операції для особливих випадків.

Арифметичні операції використовуються в математичних виразах. Арифметичні операції є близькими до операцій алгебри.

Арифметичні операції повинні мати числовий тип . Детальний опис типів даних можна прочитати тут .

Арифметичні операції можна виконувати також над типом char .

Забороняється виконувати арифметичні операції над логічними типами даних (boolean). Операція інкремента ' ++ ' збільшує значення операнда на 1.

Операція декремента ' -- ' зменшує значення операнда на 1.

Відмінністю між префіксною і постфіксною формами застосування проявляється у випадках, коли ці операції є частиною більш складного вираження.

У префіксній формі (++ a або -a) послідовність виконання операцій наступна:

збільшується (зменшується) значення операнда a на 1;

результуюче значення операнда a використовується у виразі.

У постфіксній формі (a ++ або a-) послідовність виконання операцій наступна:

попереднє значення операнда a використовується в вираженні;

збільшується (зменшується) значення операнда a на 1.

1. Java

Java — об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. З 2009 року мовою займається компанія «Oracle», яка того року придбала «Sun Microsystems». В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

«Oracle» надає компілятор Java та віртуальну машину Java, які задовольняють специфікації Java Community Process, під ліцензією GNU General Public License.

Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині. Передусім Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням, що в порівнянні, наприклад, з C++ зменшує швидкість роботи програм. За необхідності таких дій Java дозволяє викликати підпрограми, написані іншими мовами програмування.

Java вплинула на розвиток J++, що розроблялась компанією «Microsoft». Роботу над J++ було зупинено через судовий позов «Sun Microsystems», оскільки ця мова програмування була модифікацією Java. Пізніше в новій платформі «Microsoft» .NET випустили J#, щоб полегшити міграцію програмістів J++ або Java на нову платформу. З часом нова мова програмування C# стала основною мовою платформи, перейнявши багато чого з Java. J# востаннє включався в версію Microsoft Visual Studio 2005. Мова сценаріїв JavaScript має схожу із Java назву і синтаксис, але не пов'язана із Java.

1.1 Віртуальна машина Java

Віртуальна машина Java (англ. Java Virtual Machine; JVM) — набір комп'ютерних програм та структур даних, що використовують модель віртуальної машини для виконання інших комп'ютерних програм чи скриптів.

JVM використовує байт-код Java, який як правило, але не завжди генерується з вихідних кодів мови програмування Java; віртуальну машину також застосовують для виконання коду, згенерованого з інших мов програмування. Наприклад, вихідний код Ada можна скомпілювати у Java байт-код.

Віртуальна машина Java — основний компонент Java платформи. JVM доступна для всіх основних сучасних платформ, тому про програми, що скомпільовані у Java байткод теоретично можна сказати «Написано один раз, працює скрізь»

1.2. Історія Java

Мова програмування Java зародилася в 1991 р. в лабораторіях компанії Sun Microsystems. Розробку проєкту започаткував Джеймс Ґослінґ, сам проєкт мав назву «Green» (Зелений). Створення першої робочої версії, яка мала назву «Oak» (дуб), зайняло 18 місяців. Оскільки виявилось, що ім'я Oak уже використовувалось іншою фірмою, то в результаті тривалих суперечок навколо назви нової мови з-поміж ряду запропонованих було вибрано назву Java, у 1995 р. мову було офіційно перейменовано.

Головним мотивом створення Java була потреба в мові програмування, яка б не залежала від платформи (тобто від архітектури) і яку можна було б використовувати для створення програмного забезпечення, що вбудовується в різноманітні побутові електронні прилади, такі як мобільні засоби зв'язку, пристрої дистанційного керування тощо.

Досить скоро майже всі найпопулярніші тогочасні веб-оглядачі отримали можливість запускати «безпечні» для системи Java-аплети всередині веб-сторінок. У грудні 1998 р. Sun Microsystems випустила Java 2 (спершу під назвою J2SE 1.2), де було реалізовано декілька конфігурацій для різних типів платформ. Наприклад, J2EE призначалася для створення корпоративних застосунків, а значно урізана J2ME для приладів з обмеженими ресурсами, таких як мобільні телефони. У 2006

році в маркетингових цілях версії J2 було перейменовано у Java EE, Java ME та Java SE відповідно.

13 листопада 2006 року Sun випустили більшу частину Java як вільне та відкрите програмне забезпечення згідно з умовами GNU General Public License (GPL). 8 травня 2007 корпорація закінчила процес, в результаті якого всі початкові коди Java були випущенні під GPL, за винятком невеликої частини коду, на який Sun не мала авторського права.

Період становлення Java збігся у часі з розквітом міжнародної інформаційної служби World Wide Web. Ця обставина відіграла вирішальну роль у майбутньому Java, оскільки Web теж вимагала платформо-незалежних програм. Як наслідок, були зміщені акценти в розробці Sun з побутової електроніки на програмування для Інтернет.

1.3 Історія версій Java

Мова програмування Java мала декілька змін починаючи з JDK 1.0, а також багато доповнень в класах та пакетах стандартної бібліотеки. Починаючи з J2SE розвитком Java управляють Java Community Proccess (JCP), які використовують Java Specification Requests (JSRs), щоб запропоновувати та вказувати доповнення та зміни у платформі Java. Мова визначається специфікацією JLS, змінами у JLS управляють JSR 901.

В доповнення до змін мови, найбільш драматичні зміни відбувалися у Java Class Library протягом багатьох років. Бібліотека виросла з декількох сотень класів в JDK 1.0 до більш ніж трьох тисяч в J2SE5. З'явилися нові API (Swing та Java 2D), крім цього, багато класів та методів з JDK 1.0 застаріли. Деякі програми дозволяють перетворення програм на Java з однієї версії до іншої (наприклад Java 5.0 портована до 1.4).

Після релізу Java 7, Oracle пообіцяли повернутись до дворічного циклу випуску. Тим не менше, у 2013 Oracle оголосили, що вони бажають відкласти випуск Java 8 на один рік для того, щоб виправити помилки, пов'язані з безпекою Java.

1.4. Фреймворки Java

Мова Java в даний временя бере участь у функціонуванні 134 861 сайта , включаючи ESPN, SnapDeal, Alibaba і т.д. За всю історію існування ця мова довів свою високу ефективність в розробці ПЗ.
Java дуже популярний в безлічі індустрій, включаючи науку, медицину, освіту, фінанси, урядові сегменти і не тільки. Вище на діаграмі продемонстровано відсотковий розподіл мови в різних індустріальних областях.
Java є об'єктно-орієнтованою мовою програмування, головна особливість якого полягає в тому, що розробники можуть запускати написані на ньому програми під будь-який підтримує його операційною системою.
На сьогоднішній день останньою версією продукту є Java 13, запущена у вересні 2019.
Згідно з даними індексу Toibe , отриманим на основі аналізу видачі 25 найпопулярніших пошукових систем, Java лідирує, займаючи 1 позицію в списку.
Тут представлена таблиця популярності різних мов програмування станом на листопад 2019 і 2018 року.
Те, що Java стабільно утримує 1-е місце, робить його одним з найбільш затребуваних мов в середовищі розробки софта, чому також сприяють своєчасні апдейти і запуск оновлених версій.
Але одного лише вибору цієї мови для створення вашого майбутнього програми буде недостатньо. Вам також буде потрібно підібрати найбільш підходящий фреймворк.
Це непросте завдання, яке має на увазі різнобічне вивчення всіх плюсів і мінусів з урахуванням тієї сфери індустрії, для якої планується розробка.

1: Spring
Посідає перше місце з огляду на його підвищеної зручності в розробці комплексних web-додатків, що вимагають високої якості. З його допомогою розробники з легкістю створюють програмні рішення для великих підприємств.

Фактично саме це зручність і функціональність роблять його найпопулярнішим в середовищі Java-розробки . В якості підтвердження нижче приведена наочна статистика використання різних фреймворків.
Які фреймворки ви використовуєте і використовуєте взагалі?
Розробники істотно частіше вибирають Spring MVC і Spring Boot. Найбільшою перевагою цих фреймворків є можливість від'єднання інших модулів і зосередження на одному завдяки інверсії управління (IoC).
До інших переваг цього фреймворка можна віднести повноцінну модель конфігурації, підтримку усталених і новітніх баз даних, таких як NoSQL, а також цілісність процесу розробки, підтримуючи аспектно-орієнтоване програмування. Він включає в себе такі модулі, як Spring MVC, Spring Core, Spring Boost, SpringTransaction тощо.

2: Hibernate

Цей фреймворк - ORM, об'єктно-реляційна відображення, тобто він дозволяє писати запити до сервера баз даних не на SQL, а на Java, що змінює звичний погляд на базу даних як таку. Незважаючи на те, що Hibernate не є повноцінним фреймворком, він дозволяє з легкістю конвертувати інформацію для різних баз даних.
Ця особливість також спрощує масштабування, незалежно від розміру програми і кількості його користувачів. В цілому, цей фреймворк можна охарактеризувати як швидкий, потужний, легко масштабується і настроюється.

3: Struts

Struts дозволяє розробляти для підприємств простий і зручний у використанні софт. Основною перевагою цього фреймворку виступають його портативні плагіни, які є пакетами JAR.
Модулі Hibernate і Spring в цьому випадку можуть використовуватися для об'єктно-реляційного відображення і впровадження залежностей, відповідно. Цей

фреймворк також дозволяє скоротити загальний час розробки за рахунок вдалої організації Java, JSP і Action класів.

4: Play

Його застосовують такі найбільші компанії як LinkedIn, Samsung, The Guardian, Verizon і інші, що підтверджує його безумовну надійність. До основних відмінних характеристик можна віднести високу швидкість, якість і гарну масштабованість. Інтерфейс Play є дуже простим і освоюється розробниками мобільних додатків досить швидко. Застосування ж він найчастіше знаходить в тих додатках, які вимагають регулярного створення контенту.

5: Google web Toolkit

Цей безкоштовний фреймворк застосовується для розробки клієнтської частини додатків, наприклад в Javascript. Корпорація Google широко застосовувала його в створенні великої кількості своїх сервісів, включаючи AdSense, Google Wallet, AdWords тощо.

За допомогою його кодів можуть бути також легко розроблені і налагоджені додатки Ajax. Розробники вибирають цей фреймворк при написанні комплексних програм, а його основні фішки - це крос-браузерна сумісність, зберігання історії, можливість ставити мітки і ін.

6: Grails

Цей фреймворк також є безкоштовним і досить популярний в роботі з Enterprise Java Beans. Він може застосовуватися для створення надійних, масштабованих додатків, систем управління контентом, RESTful сервісів і комерційних сайтів. Grails можна застосовувати разом з іншими технологіями Java - Spring, Hibernate, quartz, SiteMesh і контейнерами EE. Його переваги проявляються у вигляді наявності GORM, гнучких профілів, просунутої системи численних плагінів і бібліотеки відображення об'єктів.

7: Blade

Практично будь-який розробник додатків зможе розібратися в цьому фреймворку протягом дня. Java Blade був випущений в 2015 і відразу став відомий як простий і легкий інструмент. Найяскравішим його перевагою є можливість дуже швидко створювати web-додатки.

Blade відноситься до повноцінних фреймворками, що пропонують просту і ясну структуру написання коду, підтримку web jar ресурсів і плагінів. Заснований він на Java 8, завдяки чому має інтерфейс маршрутизації в стилі RESTful.

8: JavaServer Faces

Цей фреймворк був впроваджений в індустрію компанією Oracle. JSF можна використовувати для створення корпоративних додатків, нативних програм, а також в web-розробці. Головна особливість полягає в можливості легко пов'язувати рівень уявлення з кодом програми.

Його API набір застосовується для подання і управління компонентами UI. JSF також має ясну архітектуру, яка різниться між логікою програми та його поданням. Ще однією відмінністю є уявлення за допомогою XML, а не Java.

9: Vaadin

Відмінна платформа для впорядкованої розробки. Великою перевагою цього фреймворка є плавне взаємодія між сервером і браузером.

Vaadin надає доступ до DOM безпосередньо з віртуальної машини Java. В останньому релізі його розділили на дві частини і перейменували в Vaadin Flow. Однак він як і раніше є легковажним фреймворком, що здійснює комунікацію і маршрутизацію на стороні сервера.

2. ООП

Об'єктно-орієнтоване програмування — це метод програмування, заснований на поданні програми як сукупності взаємодіючих об'єктів, кожен з яких є примірником певного класу, а класи є членами певної ієрархії наслідування[6].
Програмісти спочатку пишуть клас, а на його основі під час виконання програми створюються конкретні об'єкти (екземпляри класів). На основі класів можна створювати нові, які розширюють базовий клас і таким чином створюється ієрархія класів.
На думку Алана Кея, розробника мови Smalltalk, якого вважають одним з «батьків-засновників» ООП, об'єктно-орієнтований підхід полягає в наступному наборі основних принципів:
Все є об'єктами.
Всі дії та розрахунки виконуються шляхом взаємодії (обміну даними) між об'єктами, під час якої один об'єкт потребує, щоб інший об'єкт виконав деяку дію.
Об'єкти взаємодіють, надсилаючи і отримуючи повідомлення. Повідомлення — це запит на виконання дії, доповнений набором аргументів, які можуть знадобитися під час виконання дії.
Кожен об'єкт має незалежну пам'ять, яка складається з інших об'єктів.
Кожен об'єкт є представником (екземпляром, примірником) класу, який виражає загальні властивості об'єктів.
У класі задається поведінка (функціональність) об'єкта. Таким чином усі об'єкти, які є екземплярами одного класу, можуть виконувати одні й ті ж самі дії.
Класи організовані у єдину деревоподібну структуру з загальним корінням, яка називається ієрархією успадкування. Пам'ять та поведінка, зв'язані з екземплярами деякого класу, автоматично доступні будь-якому класу, розташованому нижче в ієрархічному дереві.

Таким чином, програма є набором об'єктів, що мають стан та поведінку. Об'єкти взаємодіють використовуючи повідомлення. Будується ієрархія об'єктів: програма в цілому — це об'єкт, для виконання своїх функцій вона звертається до об'єктів що містяться у ньому, які у свою чергу виконують запит шляхом звернення до інших об'єктів програми. Звісно, щоб уникнути нескінченної рекурсії у зверненнях, на якомусь етапі об'єкт трансформує запит у повідомлення до стандартних системних об'єктів, що даються мовою та середовищем програмування. Стійкість та керованість системи забезпечуються за рахунок чіткого розподілу відповідальності об'єктів (за кожну дію відповідає певний об'єкт), однозначного означення інтерфейсів міжоб'єктної взаємодії та повної ізольованості внутрішньої структури об'єкта від зовнішнього середовища (інкапсуляції).

2.1. Історія ООП

ООП виникло в результаті розвитку ідеології процедурного програмування, де дані і підпрограми (процедури, функції) їх обробки формально не пов'язані. Для подальшого розвитку об'єктно-орієнтованого програмування велике значення мають поняття події і компоненти.

Формування КОП від ООП відбулося, так само як формування модульного від процедурного програмування: процедури сформувалися в модулі - незалежні частини коду до рівня збірки програми, так об'єкти сформувалися в компоненти - незалежні частини коду до рівня виконання програми. Взаємодія об'єктів відбувається за допомогою повідомлень. Результатом подальшого розвитку ООП, мабуть, буде агентно-орієнтоване програмування, де агенти - незалежні частини коду на рівні виконання. Взаємодія агентів відбувається за допомогою зміни середовища, в якій вони знаходяться.

Мовні конструкції, конструктивно не пов'язані безпосередньо з об'єктами, але необхідні їм для їх безпечної (виняткові ситуації, перевірки) та ефективної роботи, інкапсулюються від них в аспекти (в аспектно - орієнтованому програмуванні). Суб'єктно-орієнтоване програмування розширює поняття об'єктів шляхом

забезпечення більш уніфікованого і незалежної взаємодії об'єктів. Може бути перехідною стадією між ООП та агентним програмуванням в частині самостійної їх взаємодії.

Першою мовою програмування, в якій були запропоновані принципи об'єктної орієнтованості, була Симула. На момент своєї появи (в 1967 році), ця мова програмування запропонувала революційні ідеї: об'єкти, класи, віртуальні методи тощо, однак це все не було сприйнято сучасниками як щось грандіозне. Тим не менше, більшість концепцій були розвинені Аланом Кеєм та Деном Інгаллсом у мові Smalltalk. Саме вона стала першою широко поширеною об'єктно - орієнтованою мовою програмування.

Наразі кількість прикладних мов програмування (список мов), що реалізують об'єктно-орієнтовану парадигму, є найбільшим по відношенню до інших парадигм. В області системного програмування досі застосовується парадигма процедурного програмування, і загальноприйнятою мовою програмування є мова C. Хоча при взаємодії системного і прикладного рівнів операційних систем стали помітно впливати мови об'єктно-орієнтованого програмування. Наприклад, однією з найпоширеніших бібліотек мультиплатформового програмування є об'єктно-орієнтована бібліотека Qt, написана мовою C++.

2.2. Фундаментальні поняття ООП

В результаті дослідження Дебори Дж. Армстронг (англ. Deborah J. Armstrong) комп'ютерної літератури, що була видана протягом останніх 40 років, вдалось відокремити фундаментальні поняття (принципи), використані у переважній більшості визначень об'єктно-орієнтованого програмування. До них належить:

Клас

Клас визначає абстрактні характеристики деякої сутності, включно з характеристиками самої сутності (її атрибутами або властивостями) та діями, які

вона здатна виконувати (її поведінкою, методами або можливостями). Наприклад, клас Собака може характеризуватись рисами, притаманними всім собакам, зокрема: порода, колір хутра, здатність гавкати. Класи вносять модульність та структурованість в об'єктно-орієнтовану програму. Зазвичай клас має бути зрозумілим для не-програмістів, що знаються на предметній області, що, у свою чергу, значить, що клас повинен мати значення в контексті. Також, код реалізації класу має бути досить самодостатнім. Властивості та методи класу, разом називаються його членами.

Об'єкт

Окремий екземпляр класу (створюється після запуску програми і ініціалізації полів класу). Клас Собака відповідає всім собакам шляхом опису їхніх спільних рис; об'єкт Сірко є одним окремим собакою, окремим варіантом значень характеристик. Собака має хутро; Сірко має коричнево-біле хутро. Об'єкт Сірко є екземпляром (примірником) класу Собака. Сукупність значень атрибутів окремого об'єкта називається станом. На основі класу Собака можна, також, створити інший об'єкт Дружок, який відрізнятиметься від об'єкта Сірко своїм станом (наприклад кольором хутра). Обидва об'єкта (Сірко і Дружок) є екземплярами класу Собака.

Метод

Можливості об'єкта. Оскільки Сірко — Собака, він може гавкати. Тому гавкати() є одним із методів об'єкта Сірко. Він може мати й інші методи, зокрема: місце(), або їсти(). В межах програми, використання методу має впливати лише на один об'єкт; всі Собаки можуть гавкати, але треба щоб гавкав лише один окремий собака.

Обмін повідомленнями

«Передача даних від одного процесу іншому, або надсилання викликів методів.»

Успадкування (наслідування)

Клас може мати «підкласи», спеціалізовані, розширені версії надкласу. Можуть навіть утворюватись цілі дерева успадкування. Наприклад, клас Собака може мати підкласи Коллі, Пекінес, Вівчарка тощо. Так, Сірко може бути екземпляром класу Вівчарка. Підкласи успадковують атрибути та поведінку своїх батьківських класів, і можуть вводити свої власні. Успадкування може бути одиничне (один безпосередній батьківський клас) та множинне (кілька батьківських класів). Це залежить від вибору програміста, який реалізовує клас та мови програмування. Так, наприклад, в Java дозволене лише одинарне успадкування, а в C++ і те і інше.

Приховування інформації (інкапсуляція)

Приховування деталей про роботу класів від об'єктів, що їх використовують або надсилають їм повідомлення. Так, наприклад, клас Собака має метод гавкати(). Реалізація цього методу описує як саме повинно відбуватись гавкання (приміром, спочатку вдихнути() а потім видихнути() на обраній частоті та гучності). Петро, хазяїн пса Сірка, не повинен знати як він гавкає. Інкапсуляція досягається шляхом вказування, які класи можуть звертатися до членів об'єкта. Як наслідок, кожен об'єкт надає кожному іншому класу певний інтерфейс — члени, доступні іншим класам. Інкапсуляція потрібна для того, аби запобігти використанню користувачами інтерфейсу тих частин реалізації, які, швидше за все, будуть змінюватись. Це дасть змогу полегшити внесення змін без потреби змінювати і користувачів інтерфейсу. Наприклад, інтерфейс може гарантувати, що щенята можуть додаватись лише до об'єктів класу Собака кодом самого класу. Часто, члени класу позначаються як публічні (англ. public), захищені (англ. protected) та приватні (англ. private), визначаючи, чи доступні вони всім класам, підкласам, або лише до класу в якому їх визначено. Деякі мови програмування йдуть ще далі: Java використовує ключове слово private для обмеження доступу, що буде дозволений лише з методів того самого класу, protected — лише з методів того самого класу і його нащадків та з класів із того ж самого пакету, C# та VB.NET відкривають деякі члени лише для класів із тієї ж збірки шляхом використання

ключового слова internal (C#) або Friend (VB.NET), а Eiffel дозволяє вказувати які класи мають доступ до будь-яких членів.

Абстрагування

Спрощення складної дійсності шляхом моделювання класів, що відповідають проблемі, та використання найприйнятнішого рівня деталізації окремих аспектів проблеми. Наприклад Собака Сірко більшу частину часу може розглядатись як Собака, а коли потрібно отримати доступ до інформації специфічної для собак породи коллі — як Коллі і як Тварина (можливо, батьківський клас Собака) під час підрахунку тварин Петра.

Поліморфізм

Поліморфізм означає залежність поведінки від класу, в якому ця поведінка викликається, тобто, два або більше класів можуть реагувати по-різному на однакові повідомлення. Наприклад, якщо Собака отримує команду голос(), то у відповідь можна отримати Гав; якщо Свиня отримує команду голос (), то у відповідь можна отримати Рох-рох. На практиці - це реалізовується шляхом реалізації ряду підпрограм (функцій, процедур, методи тощо) з однаковими іменами, але з різними параметрами. Залежно від того, що передається, і вибирається відповідна підпрограма.

3. Серидовище розробки

Інтегроване середовище розробки (Integrated Development Environment, IDE) - це сукупність програмних засобів, що підтримує всі етапи розробки програмного забезпечення від написання вихідного тексту програми до її компіляції та відлагодження, і забезпечує просту і швидку взаємодію з іншими інструментальними засобами. Для того щоб визначитися з вибором будь-якого засобу розробки потрібно враховувати в першу чергу можливості вашої машини і наявність в ній достатньої кількості ресурсів, інакше робота з деякими візуальними засобами перетвориться на суцільні муки, тому що більшість середовищ вимагають наявності більш-менш сучасної конфігурації комп'ютера. По-друге, тут також враховуються ваші особисті пристрасті і переваги. Деякі сидять в звичайних текстових редакторах зі стандартним JDK в консолі і не відчувають необхідності що-небудь міняти.

4. Вектор

У Java вектор можна визначити як масив об'єктів, що збільшується. Подібно до масивів, до елементів Vector можна також отримати доступ за допомогою індексів. Часто перед програмістом постає завдання організації масиву з невідомим заздалегідь розміром. Стандартні масиви в Java не підходять для вирішення цього завдання, так як їх розмір повинен бути зазначений в початковому тексті програми явно при виділенні пам'яті.

Для організації масиву з динамічно мінливих розміром ви можете скористатися класом java.util.Vector. Такий масив може зберігати об'єкти будь-якого класу.

При створенні об'єкта класу Vector розмір відповідного масиву можна не вказувати. У цьому випадку буде створений масив з розміром, прийнятому за замовчуванням. Коли ви будете додавати в нього нові елементи, розмір масиву при необхідності буде автоматично збільшуватися.

Вибравши відповідний конструктор, програміст може задати як початковий розмір масиву, так і величину, на яку відбувається збільшення цього розміру.

За допомогою методів, визначених в класі Vector, ви можете додавати в масив нові елементи (як в кінець масиву, так і в довільну клітинку зі зрушенням інших осередків), витягувати елемент за номером комірки, виконувати пошук, видалення елементів і так далі.

4.1. Бітовий вектор

Бітовий вектор - це структура, метою якої є доступність окремих бітів. З точки зору реалізації це може бути цілочисельний масив з деякою функцією, призначеної для адресації і маніпулювання окремими бітами масиву. Для кінцевого користувача масив повинен виглядати як "string" бітів, а функції повинні мати можливість доступу до довільного N-му битку "string". У стандартній бібліотеці java є клас bitset, який представляє цю концепцію

5. Розробка програми

Розробка программи починається із створення класів BitVector (параметризований клас бітового вектору(Додаток А)) та класу Kursova(користувацький інтерфейс(Додаток Б)).

Продовжується із написання трьох конструкторів для класу BitVector із різними вхідними параметрами а точніше: конструктор який нічого не приймає, конструктор який приймає масив типу Boolean значення якого записуються в основний масив типу byte, і конструктор який приймає значення довжини масиву типу int для створення пустого масиву цієї довжини.

Потім створюються наступні методи:

subscript(int idx) – повертає нижній індекс;

position(int idx) – повертає позицію;

set(boolean bool[]) - приймає масив булевого типу і записує його у вектор;

set(int index, boolean value) – приймає індекс і булеве значення яке записує по індексу в вектор;

get(int index) – повертає значення по індексу;

convert(byte theByte) – конвертує елементи типу byte в нулі та одиниці(true false);

побітові операції AND OR XOR NOT;

front() – повертає перше значення;

back() – повертає останнє значення;

size() – повертає розмір вектору;

view() – виводить;

Equals(Object obj) – порівнює чи рівні об’єкт this із об’єктом obj;

swap() – переставляє значення з одного в інший;

num_true() – рахує кількість одиниць;

num_false() – рахує кількість нулів;

5.1. Інструкція користувача

Для використання методів спочатку потрібно створити об’єкт BitVector та Kursova потім наш об’єкт потрібно заповнити для цього можна використати метод set в купі з методом convert (bitVector.set(k.convert((byte) 31));)

Для того щоб побачити результат виконання деяких методів (Equals, front,back) потрібно записувати їх у вивід детальніше можна подивитись у Додатку Б в якому написано приклад написання кожного методу

ВИСНОВКИ

Під час написання курсової роботи я дізнався що таке бітовий вектор де його та як використовувати які в нього є методи, також я навчився самостійно його реалізовувати не досконало, але трохи підтягнув свої знання з Об’єктно-орієнтованого програмування та практичні навички з програмування. Також я дізнався що структура бітовий вектор є мало використовувана та застаріла порівняно із ArrayList. Дізнався що створення своєї структури це складно та не доцільно. Також я провів порівняння вектора та ArrayList

Вектор	**ArrayList**
Наявна з початкової версії Java (версія JDK 1.0).	Введено в Java з JDK 1.2
Vector - це застарілий клас Java.	ArrayList є частиною платформи Java Collections Framework.
Вектор досягає свого розміру вдвічі, коли досягається його потужність.	ArrayList зростає вдвічі менше, коли досягається його потужність.
Векторні методи синхронізовані.	ArrayList не синхронізований.
Vector використовує Enumerator та Iterator для обходу.	ArrayList використовує лише Iterator.
Векторні операції повільніші.	ArrayList швидше.
Вектор має розмір приросту, за допомогою якого розмір вектора можна збільшити.	ArrayList не забезпечує збільшення розміру.
Vector є потокобезпечним, що означає, що використання Vector з декількох потоків дозволено та безпечно.	ArrayList не є потокобезпечним.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Операции. Арифметические операции | BestProg [Електронний ресурс] – Режим доступу до ресурсу: https://www.bestprog.net/ru/2017/01/24/%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B8-%D0%B0%D1%80%D0%B8%D1%84%D0%BC%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5-%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B8
2. Топ-9 фреймворков Java в 2020 году [Електронний ресурс] – Режим доступу до ресурсу: https://nuancesprog.ru/p/5857/
3. Фреймворки для Java [Електронний ресурс] – Режим доступу до ресурсу: https://javastudy.ru/frameworks/
4. Java [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Java
5. Об'єкт (програмування) [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)
6. Об'єктно-орієнтоване програмування [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F
7. Історія версій Java [Електронний ресурс] – Режим доступу до ресурсу:

https://uk.wikipedia.org/wiki/%D0%86%D1%81%D1%82%D0%BE%D1%80%D1%96%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D1%96%D0%B9_Java

8. Віртуальна машина Java [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%92%D1%96%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0_Java
9. Що таке бітовий вектор [Електронний ресурс] – Режим доступу до ресурсу: https://coderoad.ru/19622970/%D0%A7%D1%82%D0%BE-%D1%82%D0%B0%D0%BA%D0%BE%D0%B5-%D0%B1%D0%B8%D1%82%D0%BE%D0%B2%D1%8B%D0%B9-%D0%B2%D0%B5%D0%BA%D1%82%D0%BE%D1%80
10. Що таке Java Java [Електронний ресурс] – Режим доступу до ресурсу: https://uk.myservername.com/what-is-java-vector-java-vector-class-tutorial-with-examples#:~:text=Q%20%23%201)%20%D0%A9%D0%BE%20%D1%82%D0%B0%D0%BA%D0%B5%20%D0%B2%D0%B5%D0%BA%D1%82%D0%BE%D1%80,%D0%BE%D1%82%D1%80%D0%B8%D0%BC%D0%B0%D1%82%D0%B8%20%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%20%D0%B7%D0%B0%20%D0%B4%D0%BE%D0%BF%D0%BE%D0%BC%D0%BE%D0%B3%D0%BE%D1%8E%20%D1%96%D0%BD%D0%B4%D0%B5%D0%BA%D1%81%D1%96%D0%B2.
11. Використання класу Vector [Електронний ресурс] – Режим доступу до ресурсу: https://www.frolov-lib.ru/programming/javasamples/vol1/vol1_12/index.html
12. Побітові операції Java [Електронний ресурс] – Режим доступу до ресурсу: https://javarush.ru/groups/posts/1925-pobitovihe-operacii
13. Виключне «АБО» [Електронний ресурс] – Режим доступу до ресурсу:

https://ru.wikipedia.org/wiki/%D0%98%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B0%D1%8E%D1%89%D0%B5%D0%B5_%C2%AB%D0%B8%D0%BB%D0%B8%C2%BB

14. Побітові операції [Електронний ресурс] – Режим доступу до ресурсу: https://www.examclouds.com/ru/java/java-core-russian/pobitovie-operacii
15. Побітові операції в Java [Електронний ресурс] – Режим доступу до ресурсу: https://vertex-academy.com/tutorials/ru/ponyatie-bita-rabota-s-bitami/
16. Ссилка на масив [Електронний ресурс] – Режим доступу до ресурсу: https://ru.hexlet.io/courses/java-arrays/lessons/memory/theory_unit
17. Робота з масивами та рядками в Java [Електронний ресурс] – Режим доступу до ресурсу: http://iwanoff.inf.ua/programming_2_ua/LabTraining05.html
18. Java Vector [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com/java-vector#:~:text=Vector%20is%20like%20the%20dynamic,there%20is%20no%20size%20limit.&text=Vector%20is%20synchronized.,part%20of%20a%20collections%20framework.
19. Java BitSet [Електронний ресурс] – Режим доступу до ресурсу: https://docs.oracle.com/javase/7/docs/api/java/util/BitSet.html
20. BitVector [Електронний ресурс] – Режим доступу до ресурсу: https://flylib.com/books/en/2.300.1/bit_vectors.html
21. Bit Array [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Bit_array
22. BitVector [Електронний ресурс] – Режим доступу до ресурсу: http://www.cs.cornell.edu/courses/cs211/2007sp/Assignments/a3/doc/a3/BitVector.html
23. BitVector [Електронний ресурс] – Режим доступу до ресурсу: https://www.iro.umontreal.ca/~simardr/ssj/doc/html/umontreal/iro/lecuyer/util/BitVector.html
24. BitVector[Електронний ресурс] – Режим доступу до ресурсу:

https://dsiutils.di.unimi.it/docs/it/unimi/dsi/bits/BitVector.html

25. BitVector Питання та відповіді по бітовому векторі [Електронний ресурс] – Режим доступу до ресурсу: https://progi.pro/bitvector-t15349

ДОДАТОК А

```
public class BitVector {
    private  byte[] repn;
    private int length=0;
    private static final int BITS_PER_UNIT = 8;
    private static int subscript(int idx) {
        return idx / BITS_PER_UNIT;
    }
    private static int position(int idx) {
        return 1 << (BITS_PER_UNIT - 1 - (idx % BITS_PER_UNIT));
    }
    public BitVector(int length) throws IllegalArgumentException {
        if (length < 0) {
            throw new IllegalArgumentException("Negative length for BitArray");
        }
        this.length = length;
        repn = new byte[(length + BITS_PER_UNIT - 1)/BITS_PER_UNIT];
    }
    public BitVector(){
    }
    public BitVector(boolean[] bits) {
        length = bits.length;
        repn = new byte[(length + 7)/8];

        for (int i=0; i < length; i++) {
            set(i, bits[i]);
        }
    }
    public boolean[] convert(byte theByte){
        boolean arrayz[]= new boolean[8];
```

```
    arrayz[0] = (theByte & 0x80) != 0;
    arrayz[1] = (theByte & 0x40) != 0;
    arrayz[2] = (theByte & 0x20) != 0;
    arrayz[3] = (theByte & 0x10) != 0;
    arrayz[4] = (theByte & 0x8) != 0;
    arrayz[5] = (theByte & 0x4) != 0;
    arrayz[6] = (theByte & 0x2) != 0;
    arrayz[7] = (theByte & 0x1) != 0;
    return arrayz;
}
public boolean get(int index) {
    if (index < 0 || index >= length) {
        throw new ArrayIndexOutOfBoundsException(Integer.toString(index));
    }
    return (repn[subscript(index)] & position(index)) != 0;
}
public void set(boolean bool[]){
    for (int i=0;i<length;i++){
        int idx = subscript(i);
        int bit = position(i);
        if (bool[i]){
            repn[idx]|=bit;
        }else {
            repn[idx] &= ~bit;
        }
    }
}



public void set(int index, boolean value)
```

```
        throws ArrayIndexOutOfBoundsException {
    if (index < 0 || index >= length) {
        throw new ArrayIndexOutOfBoundsException(Integer.toString(index));
    }
    int idx = subscript(index);
    int bit = position(index);

    if (value) {
        repn[idx] |= bit;
    } else {
        repn[idx] &= ~bit;
    }
}
public void AND(BitVector bitVector){
    for (int i=0;i< bitVector.size();i++)
        this.repn[i] &= bitVector.repn[i];
    this.view();
}
public void OR(BitVector bitVector){
    for (int i=0;i< bitVector.size();i++)
        this.repn[i] |= bitVector.repn[i];
    view();
}
public void XOR(BitVector bitVector){
    for (int i=0;i< bitVector.size();i++)
        this.repn[i] ^= bitVector.repn[i];
    view();
}
public int NOT(){
    int a=0;
```

```
    for (int i=0;i< this.size();i++)
        a= ~this.repn[i];
    return a;
}
public int front(){
    return this.repn[0];
}
public int back(){
    return this.repn[repn.length-1];
}
public int size() {
    return repn.length;
}
public void view(){
    for (byte b : repn) {
        System.out.println(b);
    }
}
public String Equals(Object obj) {
    if (this == obj) return "Вони рівні";
    if (obj == null || !(obj instanceof BitVector)) return "Вони не рівні";
    BitVector ba = this;
    BitVector ba2 = (BitVector) obj;
    if (ba.length != length){
        System.out.println("1");
        return "Вони не рівні";
}
    for (int i = 0; i < repn.length; i += 1) {
        if (ba.repn[i] != ba2.repn[i])
            return "Вони не рівні";
```

```
    }
    return "Вони рівні";
}
public void swap(Object obj) {
    BitVector midle = this;
    BitVector ba = this;
    BitVector ba2 = (BitVector) obj;
    if (ba.length != length){
        return;
    }
    for (int i = 0; i < repn.length; i += 1) {
        midle.repn[i]=ba.repn[i];
        ba.repn[i]=ba2.repn[i];
        ba2.repn[i]= midle.repn[i];
    }
   ba.view();
    ba2.view();
}
public int num_true(boolean[]bool){
    Kursova k = new Kursova();
    int num=0;
    for (int i=0;i<this.length;i++)
        if (bool[i])
            num++;
    return num;
}
public int num_false(boolean[]bool){
    Kursova k = new Kursova();
    int num=0;
    for (int i=0;i<this.length;i++)
```

```
            if (!bool[i])
                num++;
        return num;
    }
}
```

ДОДАТОК Б

```
public class Kursova{
   public boolean[] convert(byte theByte){
      BitVector bv=new BitVector();
       return bv.convert(theByte);
   }
   public static void main(String[] args) {
       Kursova k = new Kursova();
       BitVector bitVector = new BitVector(8);
       bitVector.set(k.convert((byte) 31));
       System.out.println(bitVector.num_false(k.convert((byte) 31)));
       bitVector.view();

       BitVector bitVector2 = new BitVector(8);
       bitVector2.set(k.convert((byte) 58));
       bitVector2.view();
       System.out.println(bitVector.Equals(bitVector2));
       bitVector.AND(bitVector2);
       System.out.println(bitVector.back());
       bitVector.swap(bitVector2);
       bitVector.set(3,true);
       bitVector.set(2,true);
   }
}
```