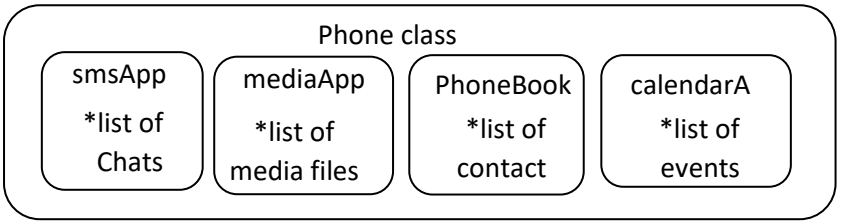
תרגיל 2 הנדסת תוכנה-קבוצה 9

נעה קוהן-315243501, מוריה ויצמן -206607459 ושירה לנדאו 208275248

בתרגיל מימשנו סביבת פלאפון, ובה 4 אפליקציות- אנשי קשר, יומן, הודעות ומדיה.

הפרויקט כולל מחלקה בשם phone שמכילה בתוכה 4 אובייקטים עבור 4 אפליקציות-

1. אנשי קשר
2. SMS
3. מדיה
4. יומן



כל אחת מהאפליקציות מכילה מבנה נתונים המחזיק אובייקטים המתאימים לאפליקציה, וכן פונקציות המטפלות בכל אחת מהפונקציונליות הנדרשת עבור אותה אפליקציה.
הגישה הארכיטקטונית שלנו בתכנון המערכת הייתה בעדיפות פונקציונלית להיררכיה נמוכה ובמודולריות מקסימלית, כלומר כל מחלקה אחראית לטפל ככל יכולתה בפונקציונליות הנדרשת (כי שיפורט בהמשך). כל מחלקה תקרא לפונקציות הממומשות בהיררכיה נמוכה יותר אם קיימות, ותממש אותן אם אינן קיימות. נראה גישה זו בכל אחת מהפונקציות:

מחלקת Phone - המחלקה הכללית:

מחלקה זו יוצרת 4 אובייקטים עבור 4 אפליקציות כפי שתואר לעיל וממשת את התפריט הראשי של הפלאפון דרכו ניתן להגיע לכל מחלקה (בעזרת מנגנון switch-case). בנוסף, מחלקה זו מכילה את מנגנון הswitch-case של התפריט של אפליקציות ההודעות והיומן מאחר והן מכילות פונקציות שבדיקת תקינות הקלט בהן דורשת סנכרון של 2 אפליקציות שונות (לדוג'- בדיקת קיום איש קשר בספר טלפונים לצורך הכנסת אירוע חדש במחלקת יומן).

```
public class Phone {
    ////fields:
    private SmsApp sms;
    private MediaApp media;
    private CalendarApp calendar;
    private PhoneBookApp phoneBook;
    private static Scanner input = new Scanner(System.in); // create an object of Scanner
    //constructors:
    public Phone();
    //methods:
    public void PhoneMenu() ;
    public void SmsApp_menu() ;
    public void CalendarApp_menu();
    //help methods:
    public String get_and_validate_contact();
}
```

מימוש האפליקציות:

1. אפליקציית אנשי קשר ContactApp:

אותו מימוש מתרגיל 1, עם השינויים הנדרשים- שינוי של פונקציית sort , הוספת בדיקה שהשם אינו קיים במערכת לפני הוספת אנשי קשר וכן התאמת התפריט לפורמט של אפליקציה (ירוץ רק כאשר הפונקציה תיקרא).

המחלקות הקשורות לאפליקציה זו:

Class PhoneBook:

```java
public class PhoneBookApp {
    //fields:
    private ArrayList<Contact> contacts;
    private static Scanner input; // create an object of Scanner
    //constructor:
    public PhoneBookApp() ;
    //getter and setter:
    public ArrayList<Contact> getContacts()
    public void setContacts(ArrayList<Contact> contacts);
    //methods:
    public static void print_menu() ;
    public void menu();
    public void add_contact() ; //1
    public void remove_contact() ; //2
    public void print_book() ; //3
    public void find_contact() ; //4
    public void sort_by_name() ; //5
    public void sort_by_number() ; //6
    public void remove_duplicate() ; //7
    public void invert_order() ; //8
    public void save_in_file() ; //9
    public void import_from_file() ; //10
    public boolean contatIsExist(String name) ;
    public boolean numIsValid(String number) ;
    public void add_contact_ahead(String  add_name, String add_num) ;
```

Class Contact:

```java
public class Contact {
    //fields:
    private String name;
    private String number;
    //constructor:
    public Contact(String new_name, String new_number) ;
    //getters & setter:
    public String get_name();
    public String get_number() ;
    public void set_contact(String new_name, String new_number) ;
    public static Comparator<Contact> ContNameComparator = new Comparator<Contact>() ;
    /*Comparator for sorting a list by Contact Number*/
    public static Comparator<Contact> ContNumberComparator = new Comparator<Contact>() ;
   //overwrite the function equals and hashCode
   //to enable comparison between two Contact objects
   @Override
   public boolean equals(Object obj);
   public int hashCode() ;
```

2. אפליקציית ההודעות - SmsApp :

קיימת מחלקת chat המייצרת אובייקטים עם שדה שם ושדה של התכתבות, ומכילה פונקציונליות של הוספת הודעה, מחיקת התכתבות, הדפסת שיחות

אפליקציית ההודעות מכילה רשימה של שיחות עם אנשי קשר, ומכילה פונקציונליות של הדפסת שיחות עם איש קשר מסוים, מחיקת התכתבות עם איש קשר, הוספת הודעה לאיש קשר- כולן מתבסוסות על הפונקציונליות הקיימת במחלקת chat. בנוסף קיימת פונקציונליות של --- הרלוונטית רק לאפליקציה ולכן ממומשת בה.

אפליקציית ההודעות תלויה באפליקציית אנשי הקשר, ולכן התפריט בו בוחרים את הפונקציה מתוך האפליקציה ממומש במחלקת Phone, כאשר לאחר שהמשתמש בוחר את הפונקציה הרצויה לו, אם יש צורך לבצע בדיקות הן מבוצעות קודם בתוך הphone ורק לאחר בדיקת תקינות הקלט נקראת הפונקציה הרלוונטית מתוך מחלקת smsApp.

המחלקות הקשורות לאפליקציה זו:

Class Chat

```java
public class Chat {
    //fields:
    private String name;
    private ArrayList<String> massages;
    //constructor:
    public Chat (String name, String new_massage) ;
    //getter
    public String getName() ;
    //methods:
    public void add_massage(String new_massage) ;
    public boolean IsContain(String massage) ;
    public void print_chat() ;
}
```

Class SmsApp

```java
public class SmsApp {
    //fields:
    private ArrayList<Chat> chats;
    private static Scanner input; // create an object of Scanner
    //constructor:
    public SmsApp() ;
    //methods:
    public void print_menu() ;
    public void add_massage(String name) ;
    public int find_chat_by_name(String name) ;
    public void delete_chat(String name) ;
    public void print_chat(String name) ;
    public void print_all_chats() ;
    public void find_phrase(String phrase) ;
}
```

3. אפליקציית היומן – CalendarApp:

באפליקציה זו נדרשנו לממש יומן של 30 יום שמכיל שני סוגי אירועים - פגישה עם איש קשר ואירוע שלא כולל פגישה. כדי לעשות זאת יצרנו מחלקה אבסטרקטית של אובייקט מסוג event ושתי מחלקות RegEvent ו-MeetingEvent שיורשות ממחלקה זו. צורה זו של בניית מחלקות האירועים אפשרה לנו ליצור באפליקציית היומן מבנה נתונים של רשימה מקושרת (linkedList) המכיל אובייקטים מסוג Event ודבר זה מאפשר לשמור אובייקטים מ-2 המחלקות היורשות במבנה נתונים אחד (פולימורפיזם). בחרנו להשתמש במחלקה אבסטרקטית ולא בממשק מאחר וזו המחלקה היחידה שהמחלקות יורשות (אין בעיה של הורשה מרובה) וכן חלק מהפונ' ממומשות במחלקה

האבסטרקית דבר שלא היה אפשרי אם היינו משתמשים בinteface. אחד השדות במחלקת event הוא אובייקט מסוג NewDate- מחלקה שיצרנו היורשת ממחלקת Date . יצרנו מחלקה זו כדי לממש את הפונקציות שהוסרו לשימוש ממחלקת Date המקורית – לדוג' getDAY,getYear וכו'...

המחלקות הקשורות לאפליקציה זו:

Class CalendarApp:

```
public class CalendarApp {

    private LinkedList<Event> calendar;
    private static Scanner input; // create an object of Scanner

    //constructor:
    public CalendarApp();
    //methods:
    public void add_event(String name);
    public void remove_event(String name);
    public boolean deleted_e(Event new_event)
    public void menu() ;
    public void show_events_of_the_day();
    public void print_meeting_with_contact(String contact);
    public void collision_detection() ;
    public NewDate Create_Date() ;
    //helper methods:
    public boolean CheckIsValidDate(int year,int month,int day);
    public boolean IsSameDate(NewDate d1,NewDate d2) ;
```

Class abstract Event

```
public abstract class Event implements Comparable<Event>{
    //fields:
    protected NewDate date;
    protected long duration; // 1-60 minutes
    protected int type;// 0 for meeting 1 for event

    public Event(NewDate date, int duration) ;
    //methods:
    public void setDate(NewDate dateToSet) ;
    public long getDuration();
    public void setDuration(int minuetes) ;
    public NewDate getDate() ;

    @Override
    public int compareTo(Event other) ;
    public String toString() ;
    public abstract boolean equals(Object obj);
```

Class RegEvent

```
public class RegEvent extends Event {
    //fields
    private String description;

    public RegEvent(NewDate date,int duration,String des);

    //methods:
    public String getDescription();
    public void setDescription(String setDes) ;
    @Override
    public String toString() ;
    public boolean equals(Object o);
```

Class meetingEvent

```
public class MeetingEvent extends Event {
    //fields
    private String contactName; //name or contact

    public MeetingEvent(NewDate date,int duration,String contact) ;
    //methods:
    public String getContact() ;
    public void setContact(String contact) ;
    @Override
    public String toString();
    public boolean equals(Object o);
```

Class NewDate:

```
public class NewDate extends Date {

    private static final long serialVersionUID = 1L;
    //fields:
    private int month;
    private int day;
    private int hour;
    private int minute;
    private int year;
    //constructors:
    public NewDate();
    public NewDate(long date) ;
    public NewDate(int year, int month, int day, int hour, int minute);
    public NewDate(int year, int month, int day) ;
    //getters and setters
    public int getYear() ;
    public void setYear(int year) ;
    public int getMonth();
    public void setMonth(int month) ;
    public int getDay() ;
    public void setDay(int day);
    public int getHour() ;
    public void setHour(int hour) ;
    public int getMinute() ;
    public void setMinute(int minute) ;

    @Override
    public boolean equals(Object o) ;
    public String toString() ;
```

4. אפליקציית מדיה- MediaApp:

באפליקציה זו המימוש כולו הוא בתוך האפליקציה עצמה (כפונקציה השייכת למחלקה) ומחלקת Phoneמ תקרא ישירות לפונקציית menu של האפליקציה שתנהל בעצמה את פעילות האפליקציה.
יצרנו מחלקה media והשתמשנו במבנה נתונים arraylist המכיל אובייקטים מסוג media

המחלקות הקשורות לאפליקציה זו:

Class MediaApp

```
public class MediaApp {
//fields:
    private ArrayList<Media> mediaList;
    private static Scanner input;
//constractors:
public MediaApp() ;
//methods:
public ArrayList<Media> getMediaList() ;
public void setMediaList(ArrayList<Media> mediaList) ;
public void menu() ;
public void AddMedia() ;
public void PlayMediaByName() ;
public void PlayAll() ;
```

Class Media

```
public class Media {
    //fields:
    private String name;
    private double length;
    private String type;
    //construtor:
    public Media(String type, String name,double length) ;
    //nethods:
    public String GetName() ;
    public double GetLength() ;
    public String GetType() ;
}
```

בדיקות קלט וטיפול בחריגות-

בכל מקום בו המשתמש נדרש להכניס קלט ישנה בדיקת תקינות לקלט. הבדיקה נעשית באופן הבא- אם הקלט עומד בתנאים (כל מקום לפי הדרישות וסוג המשתנה הנקלט) הפונקציה תמשיך ללא בעיה.

במידה והקלט אינו תקין- תודפס הודעת שגיאה למשתמש על כך שישנה טעות. אם כבר נוצר אובייקט – לאחר הופעת ההודעה המשתמש יתבקש להזין נתונים חדשים עד שיזין קלט תקין.

במידה ולא נוצר אובייקט- לאחר הודעת השגיאה הפונקציה תחזור לתפריט האפליקציה והמשתמש יצטרך להיכנס מחדש אם רוצה לתקן את הקלט.

כמו כן, הנחנו כי המשתמש מכנים קלט מtype תקין (לדוג עבור הכנסת טיפוס מסוג int המשתמש אכן יכניס טיפוס כזה)

<u>הרצה בפועל:</u> יצרנו קובץ main אשר יוצר אובייקט מסוג phone וקורא לפונקצית התפריט במחלקת phone .

```
package ex2;
public class MainProgram {
    public static void main(String[] args) {
        Phone myphone = new Phone();
        myphone.PhoneMenu();
        System.out.println("GoodBay");
    }

}
```