**Vehicle Dealership**

בפרויקט זה תממשו חברה הסוחרת בכלי רכב.

המחלקה הראשית היא VehicleDealership אשר דרכה יתבצעו כל הפעולות בmain.

מוכנים כבר אובייקטים במיין על מנת לבצע פעולות כגון קניית רכב, הצגת הקולקציה, הוספת עובדים, מכירת רכבים, הצגת העובד הטוב ביותר וכדומה.

על מנת ליצור קצת עניין בכל ריצה וריצה, פונקציות ההוספה יקבלו מערך (מספיק גדול) וישלפו בצורה אקראית ממנו עובדים ורכבים שונים.

במסמך זה, אנו נסביר **על כל פונקציה בנפרד** על מנת שיהיה ברור מה המערכת חייבת לעשות.

**קיימות 10 מחלקות אשר עליכם לממש**, תחילה על כל מחלקה נציג הסבר קצר, נראה את כל הmembers שלה ומה משמעותם, נציג ברשימה את הפונקציות אשר תדרשו לממש ולבסוף נרחיב על כל פונקציה ופונקציה אשר תצטרכו לממש מה היא צריכה לעשות ובאיזה קריטריונים עליה לעמוד.

**אנו נציין בכל מחלקה מתי קיימת הקצאה דינאמית אך נתאר רק את פונקציית הבנאי הראשי. לא נרשום את חמשת הפונקציות הנוספות שעליכם לממש בכל מחלקה (מלבד איסור השכפול במחלקה VehicleDealership)**

**Class VehicleDealership**

הקלאס הראשי, דרכו יתבצעו כל הפעולות של המערכת.

המחלקה תשלוט על הוספה והסרה של כלי רכב ומוכרים בצורה אקראית מתוך המאגר הנתון, להדפיס את הקולקציה הקיימת ואת הרווח החודשי של החברה, להחזיר האם קיימים לפחות 2 רכבים זהים בחברה ולהחזיר את העובד הטוב ביותר ואת כלי הרכב הטוב ביותר על פי הקריטריונים שנרחיב עליהם במחלקות הרלוונטיות.

**נאסור על שכפול מחלקה זו.**

**הmembers של המחלקה הם: (private)**

char name[MAX\_DS\_NAME] - שם החברה בגודל קבוע

double monthlyProfit - הרווח החודשי של החברה

Building& place – בניין החברה אשר מחזיק גם מיקום. גודל הבניין הוא זה שיחליט לנו על מספר הרכבים המקסימלי אשר אנו יכולים להחזיק.

Salesman\*\* salesmanArr – מערך של מערכים של המוכרים שלנו, נרחיב בהמשך

unsigned salesmanCount - מספר המוכרים הנוכחי

unsigned maxSalesman - מספר המוכרים המקסימלי

Vehicle\* vehicleArr - מערך של כלי רכב (פולימורפיזם)

unsigned vehicleCount - מספר הרכבים שברשותינו

unsigned maxVehicles - מספר הרכבים המקסימליים, תלויים בערך vehicleCapacity של הבניין אשר אנו שוכרים.

יש לממש את ה methods הבאים:

1. בנאי המקבל את כל הנתונים לאיתחול.
2. פונקציית get לרווח החודשי
3. פונקציית הוספת כלי רכב
4. פונקציית הוספת מוכר
5. פונקציית הוספת כלי רכב אקראיים (מערך הרכבים המוכן במיין גדול מספיק בשביל התרגיל)

הפונקציה תעשה שימוש בפונקציית הוספת כלי רכב (פונקציה מספר 3)

1. פונקציית הוספת מוכרים אקראיים מהמערך הנתון (מערך המוכרים המוכן במיין גדול מספיק)

הפונקציה תעשה שימוש בפונקציית הוספת מוכר (פונקציה מספר 4)

1. פונקצייה להצגת קולקציית הרכבים שברשותינו
2. פונקציה אשר בודקת האם קיימים 2 רכבים "זהים" ברשותינו, נסביר בהמשך על האופרטור == במחלקה Vehicle ומהי דרישת השיוון.
3. פונקציית מכירת רכב
4. פונקציית מכירת כל הקולקציה, עושה שימוש בפונקציית מכירת הרכב (פונקציה מספר 9)
5. 2 פונקציות המחזירות את העובד הטוב ביותר ואת הרכב הטוב ביותר, נרחיב במשך על האופרטורים הנדרשים ועל מה נחשב יותר טוב.

**Public methods (class VehicleDealersip):**

**VehicleDealership(const char\* name, Building& place, int maxWorkers);**

הבנאי הראשי של המחלקה. מקבל את שם החברה, בניין, ומספר העובדים המקסימלי, ומאתחל את החברה שלנו.

קיימות כאן הקצאות דינאמיות. יש לשים לב שהמערך של העובדים הוא מערך של פוינטרים, בהמשך יואתחלו. מערך כלי הרכב הוא מערך רגיל אשר גודלו יתקבל מגודל הבניין. (vehicleCapacity)

כמתואר בחלק של הmembers לעיל.

יש לזכור לעדכן את monthlyProfit משום שהבניין עולה כסף

**bool addRandomVehicles(Vehicle\*\* vehicleArr);**

פונקציה זו מקבלת מערך מהמיין ותשתמש בפונקציה buyVehicle על מנת לאתחל את מערך כלי הרכב. יש לשים לב כי יש לשמור מערך (בגודל קבוע) של העתקים מוקצים דינאמית.

הפונקציה בוחרת באקראיות מספר ומכניסה אותו על ידי שימוש בפונקציית העזר למערך העובדים בחברה. הפונקציה מחזירה false במקרה של כישלון.

הרכבים אשר מוכנסים למערך חייבים להיות שונים זה מזה (בדיקה האם האינדקס כבר הוכנס).

**bool hireRandomSalesmen(const Salesman& workerArr);**

פונקציה זו מקבלת מערך מהמיין ותשתמש בפונקציה addSalesman על מנת לאתחל את מערך המוכרים. מערך זה גם כן באורך קבוע אך אינו כולל הקצאות דינאמיות אלא העתקה עמוקה על ידי השימוש באופרטור =

פונקציה זו בוחרת באקראי מספר ומכניסה למערך העובדים. מחזירה false במקרה של כישלון.

העובדים אשר מוכנסים למערך חייבים להיות שונים זה מזה (בדיקה האם האינדקס כבר הוכנס).

**double getMonthlyProfit() const;**

Getter רגיל, לא נרחיב.

**bool addSalesman(const Salesman& s);**

מכניסה מוכר למערך שלנו.

**יש לזכור לעדכן את הערך monthlyProfit בהתאם למשכורת העובדים.**

**מחזירה** false **במקרה של כשלון**

**bool buyVehicle(const Vehicle& v);**

מכניסה כלי רכב למערך שלנו בהקצאה דינאמית.

**יש לזכור לעדכן את הערך monthlyProfit בהתאם לעלויות הרכבים.**

פונקציה זו מעדכנת את המחיר של כל רכב לפי פונקציית setPrice אשר נרחיב עליה במחלקות של הרכבים. (נרצה להגדיל את המחיר של הרכב שהרגע קנינו).

מחזירה false במקרה של כשלון

**void showCollection() const;**

פונקציה זו עושה שימוש באופרטור >> של Vehicle ומדפיסה את נתוני כלי הרכב שברשותנו.

**bool hasIdenticalVehicle() const;**

פונקציה זו עושה שימוש באופרטור == של vehicle ובודקת האם יש 2 כלי רכב זהים במערך שלנו, 2 כלי רכב זהים כאשר יש להם את אותו צבע, אותה חברה ואותו מחיר, נזכיר זאת גם באופרטור == של המחלקה Vehicle.

**void sellCollection();**

פונקציה זו עושה שימוש בפונקציה sellVehicle.נמכור את כל כלי הרכב שברשותנו. יש לשים לב כי כלי הרכב מוקצים דינאמית.

**void sellVehicle();**

פונקציה זו מוכרת כלי רכב (מוחקת את ההקצאה) ומעדכנת את הרווח שעשינו מהמכירה.

**const Vehicle\* getBestVehicle() const;**

פונקציה זו עושה שימוש באופרטור < של Vehicle ומחזירה את הרכב היקר ביותר.

**const Salesman& getBestWorker() const;**

פונקציה זו עושה שימוש באופרטור גדול שווה =< של Salesman על מנת לבדוק מי העובד הטוב ביותר. מחזירה את העובד אשר יודפס במיין על ידי ה>> שלו.

**Abstract Class Vehicle**

**הקלאס Vehicle הינו אבסטרטי.**

**ממנו יורשים הקלאס** Boat, Car **ולבסוף** SuperBoatCar **בהורשת יהלום, כפי שניתן לראות בClass Diagram שלרשותכם.**

**עקב הורשה זו נעשה שימוש בפרויקט בפולימורפיזם. המקום היחיד אשר מחזיק את מערך כלי הרכב הוא הVehicleDealership אשר מסביר כיצד לממש את מערך זה.**

**הקלאס Vehicle, בנוסף לmembers שלו אשר נרחיב עליהם מיד, מחזיק גם 2 enums ו2 מערכים אשר עושים שימוש בenums שלנו. ניתן לפנות אליהם רק דרך הקלאס. לפניכם הערכים:**

typedef enum { sfBoat, sfCar, sfSuperBoatCar, eNofSFactor } sellFactor;

typedef enum { eWhite, eBlack, eBlue, eRed, eYellow, ePurple, eNofColor } COLOR;

const double factorArr[eNofSFactor] = { 1.4, 1.3, 2 };

const char\* colorArr[eNofColor] = { "White", "Black", "Blue", "Red", "Yellow", "Purple" };

**הmembers של המחלקה הם: (protected)**

const char\* companyName – שם הייצרן, לא בגודל קבוע ולכן **קיימת הקצאה דינאמית**.

COLOR color – צבע הרכב, יעשה שימוש במערך colorArr אשר מוגדר בתוך מחלקה זו.

float price – מחיר כלי הרכב

const float sellFactor – ערך זה הוא ערך קבוע במחלקות (אך שונה לכל כלי רכב) אשר ניתן לכל כלי רכב כאשר ניצור אותו. הערך יתקבל מהמערך factorArr אשר מופיע למעלה. כל רכב יאתחל את כל זה בעת יצירתו עם הערך המתאים לו.

**יש לשים לב כי הערך COLOR ישמש אתכם בהמשך באופרטור ההשוואה, וגם באופרטור ההדפסה >>operator.**

יש לממש את ה methods הבאים:

1. בנאי המקבל את כל הנתונים לאיתחול.
2. אופרטור << להדפסה אשר מקבל vehicle ומדפיס בפולימורפיזם כפי שלמדנו בכיתה.
3. פונקצייה toOs אשר עוזרת לאופרטור >> לבצע את ההדפסות.
4. אופרטור < אשר בודק איזה כלי רכב יותר טוב.
5. אופרטור == אשר בודק שיוון בין כלי רכב שונים.
6. גטרים למחיר, צבע והשם. יש לשים לב שאת הצבע נחזיר בתור מחרוזת לפי המערך colorArr

**Public methods (class Vehicle):**

**Vehicle(const char\* companyName, COLOR color, float price);**

הקלאס אבסטרקטי לכן רק ילדיו של Vehicle יעשו שימוש בבנאי זה.

בנאי זה מקבל את שם הייצרן, צבע ומחיר כלי הרכב. יש לשים לב כי קיימת כאן הקצאה דינאמית לשם.

**friend std::ostream& operator<<(std::ostream& out, const Vehicle& v);**

אופרטור זה ישתמש בפונקציות toOs כפי שלמדנו בכיתה על מנת להדפיס את כלי הרכב השונים שלנו.

**virtual bool operator>(const Vehicle& other) final;**

כלי רכב יותר טוב מהשני כאשר הוא יותר יקר, כמו בחיים האמיתיים.

**virtual bool operator==(const Vehicle& other) final;**

כלי רכב שווה לאחר כאשר יש להם את אותו צבע, אותה חברה ואותו מחיר.

גטרים

**virtual float getPrice() const final;**

**virtual const char\* getCompanyName() const final;**

**virtual const char\* getColor() const final;**

פונקציה זו עושה שימוש במערך colorArr אשר יחזיר את המחרוזת לפי הצבע שאנו מחזיקים.

**Protected methods (class Vehicle):**

**virtual void toOs(std::ostream& out) const;**

כפי שלמדנו בכיתה, operator<< עושה שימוש בה על מנת להדפיס.

**virtual void setPrice() = 0;**

הילדים חייבים לממש את הפונקציה הזו, פונקציה זו מכפילה את מחיר כלי הרכב בsellfactor של כלי הרכב, לכל ילד factor שונה, לפי המערך float שמוחזק בvehicle. יעשה שימוש בפונקציה זו כאשר לאחר שכלי רכב נקנה (להכפיל בקבוע המתאים את המחיר על מנת שיתקבל רווח מהרכב).

**Class Car:**

**רכב הוא חלק מVehicle ולכן יורש את תכונותיו.**

**הmembers של המחלקה הם: (private)**

int drivingSpeed – מהירות הרכב

unsigned numOfWheels; - מספר הגלגלים

יש לממש את ה methods הבאים:

1. בנאי המקבל את כל הנתונים של Vehicle אך גם מהירות ומספר גלגלים.
2. גטרים למהירות ומספר גלגלים.
3. הפונקציה toOs העוזרת בהדפסה
4. הפונקציה setPrice שתיעזרו בה בעת קניית רכב בחברה. ראו הסבר במחלקה Vehicle

**Public methods (class Car):**

**Car(const char\* companyName, COLOR color, float price, int drivingSpeed, unsigned numOfWheels);**

בנאי, מקבל את כל הערכים אשר Vehicle מקבל ועושה שימוש בקונסטרטור שלו. בנוסף לכך, מאתחל את הערך sellFactor על פי המערך factorArr. למשל במקרה זה הקלאס Car ישתמש בערך sfCar על מנת לקבל את הערך שלו מהמערך factorArr.

גטרים, בשימוש בהדפסה ובהשוואות

**virtual int getNumOfWheels() const;**

**virtual int getSpeed() const;**

**Protected methods (class Car):**

**virtual void toOs(std::ostream& out) const;**

ישורשרו כאן הערכים של כלי הרכב כפי שנלמד בכיתה על הדפסת פולימורפיזם.

**virtual void setPrice();**

בדומה להסבר בVehicle.

**Class Boat:**

**סירה היא חלק מVehicle ולכן יורשת את תכונותיו.**

**הmembers של המחלקה הם: (private)**

int sailingSpeed; - מהירות הסירה.

int numOfSails; - מספר המפרשים של הסירה.

יש לממש את ה methods הבאים:

1. בנאי המקבל את כל הנתונים של Vehicle אך גם מהירות ומספר מפרשים.
2. גטרים למהירות ומספר מפרשים.
3. הפונקציה toOs העוזרת בהדפסה.
4. הפונקציה setPrice שתיעזרו בה בעת קניית רכב בחברה. ראו הסבר במחלקה Vehicle

**Public methods (class Boat):**

**Boat(const char\* companyName, COLOR color, float price, int sailingSpeed, int numOfSails);**

בנאי, מקבל את כל הערכים אשר Vehicle מקבל ועושה שימוש בקונסטרטור שלו. בנוסף לכך, מאתחל את הערך sellFactor על פי המערך factorArr כפי הסברנו בבנאי של Car.

גטרים, בשימוש בהדפסה ובהשוואות

**virtual int getNumOfSails() const;**

**virtual int getSpeed() const;**

**Protected methods (class Boat):**

**virtual void toOs(std::ostream& out) const;**

ישורשרו כאן הערכים של כלי הרכב כפי שנלמד בכיתה על הדפסת פולימורפיזם.

**virtual void setPrice();**

בדומה להסבר בVehicle

**Class SuperBoatCar:**

**סופר סירה-רכב יורש בירושת יהלום את Vehicle, Car & Boat ולכן מקבל את כל תכונותיהם, נרחיב בהמשך.**

**הmembers של המחלקה הם: (private)**

float static turbo; - כלי רכב זה יכול לעשות טורבו. נעשה שימוש בערך זה רק בפונקציה getMaxSpeed כאשר נרצה להדפיס בעזרת toOs את המהירות המקסימלי שלנו.

יש לממש את ה methods הבאים:

1. בנאי המקבל את כל הנתונים של Vehicle אך גם מהירות בים, ביבשה, מספר מפרשים ומספר גלגלים.
2. גטרים למהירות, המהירות היא המקסימלית מבין המהירות בים וביבשה. (ללא טורבו)
3. הפונקציה toOs העוזרת בהדפסה, היא תדפיס גם את המהירות המקסימלים עם טורבו על ידי שימוש בפונקציה getMaxSpeed
4. הפונקציה setPrice שתיעזרו בה בעת קניית רכב בחברה. ראו הסבר במחלקה Vehicle

**Public methods (class SuperBoatCar):**

**SuperBoatCar(const char\* companyName, COLOR color, float price, int drivingSpeed, unsigned numOfWheels, int sailingSpeed, unsigned numOfSails);**

בנאי, מקבל את כל הערכים אשר Vehicle מקבל ועושה שימוש בקונסטרטור שלו. בנוסף לכך, מאתחל את הערך sellFactor על פי המערך factorArr כפי הסברנו בבנאי של Car.

נקבל גם את ערכי שני הירושות שלנו (מהירויות שונות בים וביבשה, מספר גלגלים ומספר מפרשים).

גטרים:

**int getSpeed() const override;**

מחזיר את המקסימום מבין מהירות היבשה למהירות בים

**float getMaxSpeed() const;**

מחזיר את הכפל בין המהירות בgetSpeed וערך הטורבו שלנו

**Protected methods (class SuperBoatCar):**

**void toOs(std::ostream& out) const;**

כאן הפונקציה תדפיס גם את המהירות המקסימלים עם טורבו על ידי שימוש בפונקציה getMaxSpeed

**void setPrice() override;**

בדומה להסבר בVehicle

**Class Address:**

**מחלקה פשוטה אשר מחזיקה כתובת, כפי שנלמד בכיתה.**

**הmembers של המחלקה הם: (private)**

char\* city; - שם העיר, הקצאה דינאמית

char\* street; - שם הרחוב, הקצאה דינאמית

unsigned houseNum; - מספר הבית

יש לממש את ה methods הבאים:

1. בנאי אשר מקבל שם עיר, רחוב ומספר בית ומאתחל את הערכים.
2. האופרטור << אשר יאתחל כתובת
3. גטרים

**Public methods (class Address):**

**Address(const char\* city, const char\* street, unsigned houseNum);**

בנאי רגיל אשר מקצה דינאמית את מחרוזות העיר והרחוב ומאתחל את האובייקט.

**friend std::istream& operator>>(std::istream& in, Address& date);**

אופרטור אתחול של כתובת.

גטרים להדפסות:

**const char\* getCity() const;**

**const char\* getStreet() const;**

**unsigned getHouseNum() const;**

**Class Building:**

**בניין יורש מכתובת, לכן קיימים לו גם ערכי המיקום שלו. יהיה שימוש בבניין במחלקה הראשית על מנת לדעת דברים כמו מיקום, כמות רכבים ועדכוני הוצאות.**

**הmembers של המחלקה הם: (private)**

int vehicleCapacity; - כמה רכבים יכולים להיכנס בבניין לכל היותר

int cost; - כמה הבניין עולה לחודש (שכירות)

יש לממש את ה methods הבאים:

1. 2 בנאים אשר נרחיב עליהם
2. גטרים

**Public methods (class Building):**

**Building(const char\* city, const char\* street, unsigned houseNum, int vehicleCapacity, int cost);**

**Building(Address& loc, int vehicleCapacity, int cost);**

בנאי אחד שמקבל את כל הערכים של הAddress ואחד שמקבל מוכן address, יעשה בזה שימוש בmain מאחר וaddress מממש את האופרטור <<

בנוסף מקבלים גם את מספר הרכבים המקסימליים שיכול להכיל ואת השכירות.

גטרים אשר נשתמש בהם בבנאי של הDealership על מנת לעדכן ערכים בחברה כפי שכבר ציינו במחלקה **VehicleDealership**

**unsigned getCapacity() const;**

**unsigned getCost() const;**

**Class Date:**

**מחלקה פשוטה של תאריך ללא הקצאות דינאמיות, כפי שנלמד בכיתה.**

**הmembers של המחלקה הם: (private)**

**int year; - שנה**

**unsigned month; - חודש**

**unsigned day; - יום**

יש לממש את ה methods הבאים:

1. בנאי ברירת מחדל עם הערך 1995.7.27
2. אופרטור >> להדפסת תאריך
3. גטרים

**Public methods (class Date):**

**Date(int year, unsigned month, unsigned day);**

בנאי כפי שנלמד בכיתה עם ערכי ברירת מחדל שציינו בסעיף 1.

**friend std::ostream& operator<<(std::ostream& out, const Date& date);**

יעשה שימוש באופרטור זו כאשר יהיה צורך להדפיס תאריך של עובדים.

גטרים:

**int getYear() const;**

**unsigned getMonth() const;**

**unsigned getDay() const;**

**Class Worker:**

**מחלקה המתארת עובד, קיימת כאן הורשה על מנת שמעובד נוכל לייצר מוכר אשר אותו נחזיק במחלקה הראשית.**

**הmembers של המחלקה הם: (Protected)**

static unsigned id; - מספר זהות אשר כבר הותחל במערכת ל1000

unsigned workerID; - מספר הזהות אשר יחולק לכל אדם שיהיה ייחודי

char\* name; - שם, הקצאה דינאמית

unsigned numOfAddresses; - מספר הכתובות שיש לאדם (יכול גם להיות חסר בית)

Address\* address; - מערך של כתובות, **לא** מוקצה דינאמית.

Date birthday; - יום ההולדת

int salary; - משכורת העובד

יש לממש את ה methods הבאים:

1. בנאי לייצור עובד
2. אופרטור >> להדפסה
3. גטרים

**Public methods (class Worker):**

**Worker(const char\* name, const Address\* address, Date& birthday, int salary, unsigned numOfAddresses);**

בנאי שמקבל שם ומקצה אותו דינאמית, בנוסף מחזיק גם את הכתובת בתור הקצאה דינאמית, ואת גודל ההקצאה.

**friend std::ostream& operator<<(std::ostream& out, const Worker& worker);**

משרשר לostream את ערכי הworker.

גטרים:

**int getSalary() const;**

**const Address\* getAddress() const;**

**unsigned getNumOfAddress() const;**

**const Date& getBirthday() const;**

**const char\* getName() const;**

סטר:

**void setSalary(unsigned salary);**

**Class Salesman:**

**המחלקה מוכר – יורשת מWorker את תכונותיו מאחר ומוכר הוא סוג של עובד (בפרויקט זה קיים רק סוג אחד של עובד) – ללא הקצאות דינאמיות משלה.**

Salesman **יורש את כל ערכיו של Worker.**

**הmembers של המחלקה הם: (private)**

unsigned numOfDeals; - מספר העסקאות של המוכר.

double totalSales; - כמה כסף המוכר ייצר לחברה (סכום מחירי כלי הרכב שהוא מכר).

יש לממש את ה methods הבאים:

1. בנאי לייצור מוכר
2. אופרטור >> להדפסה
3. אופרטור =< אשר משווה בין מוכרים
4. גטרים

**Public methods (class Salesman):**

**Salesman(const char\* name, const Address\* address, Date& birthday, int salary, unsigned numOfAddresses);**

בנאי רגיל, הworker הוא זה שמקצה דינאמית ולכן לא צריך לממש בקלאס הזה את ה5.

**friend std::ostream& operator<<(std::ostream& out, const Salesman& Salesman);**

משתמש ב>> של Worker ומשרשר גם את הערכים שלו כאשר נדפיס את העובדים. יש גם להדפיס את המכירה הממוצעת המחושבת על ידי **getAverageProfit.**

**bool operator>=(const Salesman& other);**

השוואת מוכרים, נשווה מוכרים על פי הTotalSales שלהם.

**double getAverageProfit() const;**

גטר אשר מחשב את מכירה הממוצעת – סכום המכירות חלקי מספר העסקאות.