Практична робота №1
Технологічний стек компанії Microsoft
Система оцінювання

№	Тема	К-ть балів
1.	*Захист принаймні одного завдання з роботи*	1
2.	Завдання 1	0,2*
3.	Завдання 2	0,7*
4.	Завдання 3	1*
5.	Завдання 4	1*
6.	Завдання 5	0,8*
7.	*Здача звіту*	0,3
	Всього	**5**

* – діє бонусна система

Завдання на практичне заняття

1. Написати програму для платформи .NET Core, яка дозволятиме вивести автора програми та часовий штамп.

2. Інтернет-провайдер має три тарифи для населення:

Тариф 1000	1000 Мб на місяць за 20 грн, позатарифно: 1Мб = 0.05 грн.
Тариф 2000	2000 Мб на місяць за 35 грн, позатарифно: 1Мб = 0.04 грн.
Тариф 5000	5000 Мб на місяць за 85 грн, позатарифно: 1 Мб = 0.02 грн.

Створіть проект .NET Core та напишіть програму, яка повинна отримувати цифру-тариф та кількість витрачених мегабайтів, а виводити підсумковий рахунок за місяць. Якщо використовується тариф 1000 або 2000, виведіть додатково інформацію про те, скільки б платили користувачі, якби перейшли на більш дорогі тарифи.
Також реалізуйте набір модульних тестів, які перевірять наступні тестові випадки

№	Тестовий випадок	Очікуваний результат
1.	Тариф 1000, витрачено 800 Мб	20грн
2.	Тариф 1000, витрачено 1200 Мб	30грн
3.	Тариф 1000, витрачено 2200 Мб	80грн
4.	Тариф 1000, витрачено 5200 Мб	230грн
5.	Тариф 2000, витрачено 800 Мб	35грн
6.	Тариф 2000, витрачено 2200 Мб	43грн
7.	Тариф 2000, витрачено 5200 Мб	163грн
8.	Тариф 5000, витрачено 800 Мб	85грн
9.	Тариф 5000, витрачено 5200 Мб	89грн

3. Створіть консольний проект .NET Framework та напишіть клас, який знаходить площу чотирикутника, утвореного чотирма містами України: Києвом, Львовом, Одесою, Харковом. Програма має отримувати GPS координати (довготу та широту) міст, знаходити відстані між ними за формулою

$$d = R \cdot \arccos(\sin x_1 \sin x_2 + \cos x_1 \cos x_2 \cos(y_1 - y_2)),$$

де R – радіус Землі (в середньому 6371,01км), а потім обчислювати площу чотирикутника як суму площ двох трикутників (кожна за формулою Герона), отриманих з'єднанням двох несусідніх міст.

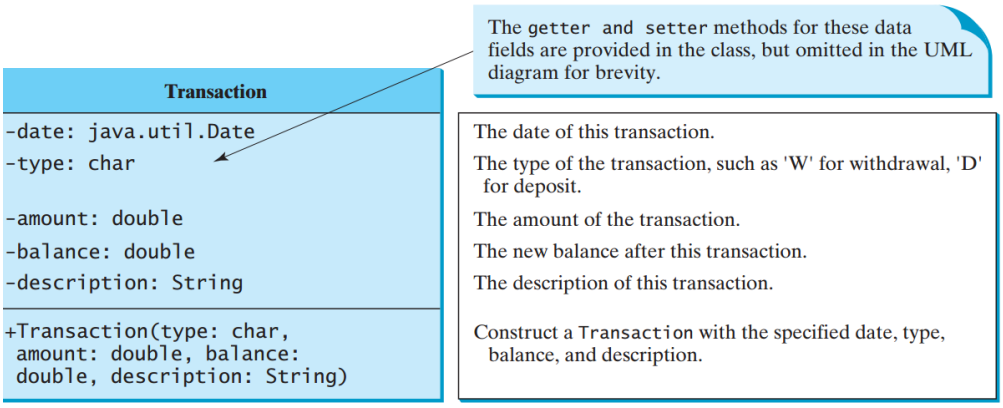
Рекомендується окремо виділити метод для знаходження відстані між двома населеними пунктами, а також виокремити метод для обчислення площі прямокутника.

Портуйте даний проект на .NET Core, як описано, наприклад, за посиланнями https://docs.microsoft.com/en-us/dotnet/core/porting/, https://www.cafe-encounter.net/p2380/migrating-net-framework-to-netcore та порівняйте структуру проектів, розміри виконуваних файлів тощо.

4. Створіть проект .NET Core та реалізуйте клас Account, який буде описувати банківський рахунок і містити:
   - Приватне цілочисельне поле id рахунку (за замовчуванням 0)
   - Приватне дробове поле balance (за замовчуванням 0)
   - Приватне дробове поле annualInterestRate, яке зберігає поточну відсоткову ставку (за замовчуванням 0). Припускаємо, що ставки для всіх рахунків однакові.
   - Приватне поле dateCreated типу Date, яке зберігає дату відкриття рахунку.
   - Безаргументний конструктор, який створює рахунок за замовчуванням.
   - Конструктор, що відкриває рахунок із заданими id та стартовим балансом.
   - Геттери/сеттери (accessor і mutator методи) для id, balance та annualInterestRate.
   - Геттер для dateCreated.
   - Метод getMonthlyInterest(), який повертає щомісячну процентну ставку для поточного балансу (не відсоток, а суму грошей).
   - Метод withdraw(), який знімає з рахунку певну кількість коштів.
   - Метод deposit() виконує внесок на рахунок.

   Створіть рахунок з ID=1122, балансом 2000 грн. і відсотковою ставкою 18,5%. Спробуйте зняти 250 грн з рахунку, а потім покласти на нього 300 грн.

   Далі спроектуйте та реалізуйте нову версію класу, що включатиме також:
   - Нове поле name типу string, яке зберігатиме ПІБ власника рахунку
   - Новий конструктор, який створює рахунок відповідно до name, id, balance.
   - Нове поле transactions типу ArrayList, яке зберігає транзакції з рахунком. Кожна транзакція описується класом Transaction, діаграма якого зображена на рисунку



- Змініть методи withdraw() та deposit() з урахуванням спискового масиву транзакцій.

Протестуйте програму, створивши рахунок з відсотковою ставкою 14%, балансом 1000 грн, id=1122, а ім'я власника – Ваше. Зробіть внески по 50, 100 та 200 грн, а потім зніміть з рахунку 40, 20 та 10 грн. Виведіть повний звіт по рахунку: ім'я власника, процентну ставку, баланс та всі транзакції.

Налаштуйте платформне мультитаргетування для проекту так, щоб існувала підтримка бібліотек .NET Standard та .NET Framework. У цьому може допомогти стаття https://www.cafe-encounter.net/p2312/multi-targetting-net-framework-and-net-core-in-a-single-project.

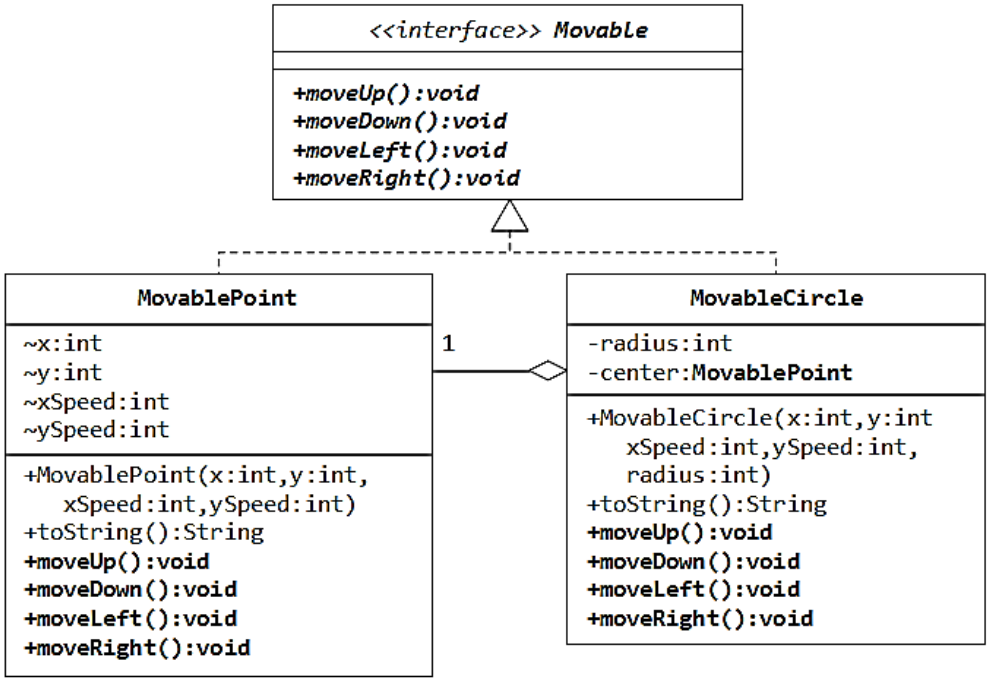
5. Припустимо, що Ви маєте набір об'єктів з подібною поведінкою: вони можуть рухатись вгору, вниз, вправо або вліво. Конкретна поведінка (як рухатись та на яку відстань) залежить від самих об'єктів. Поширеним способом моделювання такої поведінки є визначення інтерфейсу Movable з абстрактними методами moveUp(), moveDown(), moveLeft() та moveRight(). Класи, які реалізують інтерфейс Movable, будуть постачати конкретні реалізації абстрактних методів.

   Напишіть два конкретних класи MovablePoint та MovableCircle, які реалізують інтерфейс Movable відповідно до діаграми класів.

   Реалізуйте тестовий клас та спробуйте додати в нього наступні інструкції:

```
Movable m1 = new MovablePoint(5, 6, 10);     // upcast
Console.WriteLine(m1);
m1.moveLeft();
Console.WriteLine(m1);

Movable m2 = new MovableCircle(2, 1, 2, 20); // upcast
Console.WriteLine(m2);
m2.moveRight();
Console.WriteLine(m2);
```



Додайте новий клас MovableRectangle, який компонує два MovablePoints (верхній лівий та нижній правий кути) та реалізує інтерфейс Movable, як це показано на діаграмі. Переконайтесь, що обидві точки мають однакову швидкість.