**Klasa Lekarz:**

**Opis:** Klasa **Lekarz** reprezentuje lekarza. Każdy lekarz posiada harmonogram, listę zajętych terminów oraz listę odbytych wizyt. Klasa implementuje interfejs **IComparable<Lekarz>**, umożliwiając porównywanie lekarzy.

**Właściwości:**

1. **ImieLekarza**: Imię lekarza.
2. **NazwiskoLekarza**: Nazwisko lekarza.
3. **Specjalizacja(EnumSpecjalizacja)**: Specjalizacja lekarza.
4. **Harmonogram**: Harmonogram lekarza.
5. **Zajete**: Lista zajętych terminy lekarza.
6. **Odbyte\_lekarz**: Lista odbytych wizyt lekarza.

**Konstruktory:**

1. **Lekarz()**: Konstruktor domyślny inicjalizujący listy harmonogramu, zajętych terminów i odbytych wizyt.
2. **Lekarz(imieLekarza, nazwiskoLekarza, specjalizacja)**: Konstruktor parametryczny, ustawiający podstawowe informacje o lekarzu.

**Metody**:

1. **DodawanieTerminu(termin)**: Dodaje nowy termin do harmonogramu lekarza, uwzględniając dostępność i unikalność daty.
2. **PrzeniesOdbyteTerminy()**: Przenosi odbyte wizyty do listy odbytych, aktualizuje listę zajętych terminów.
3. **CompareTo(other)**: Implementacja interfejsu **IComparable**, umożliwia porównywanie lekarzy na podstawie nazwiska i imienia.
4. **WypiszLekarza()**: Zwraca informacje o lekarzu (imię, nazwisko, specjalizacja) jako tekst.
5. **WypiszHarmonogram()**: Zwraca tekstowy harmonogram lekarza, uwzględniający tylko przyszłe terminy.
6. **WypiszZaplanowaneWizytyLekarza()**: Zwraca tekstowe informacje o zajętych terminach lekarza, przenosząc odbyte wizyty.

**Klasa Termin**

**Opis:** Klasa **Termin** reprezentuje pojedynczy termin. Każdy termin ma przypisaną datę oraz informację o dostępności.

**Właściwości:**

1. **Data**: Data wizyty (dostęp do odczytu i zapisu).
2. **Dostepny**: Informacja o dostępności terminu (dostęp do odczytu i zapisu).

**Konstruktory:**

1. **Termin()**: Konstruktor domyślny, inicjalizuje obiekt Termin bez przypisanej daty i dostępności.
2. **Termin(data, dostepny)**: Konstruktor parametryczny, ustawiający datę i dostępność terminu.

**Metody:**

1. **ToString()**: Przesłonięta metoda zwracająca tekstową reprezentację terminu.

**Klasa Zwierze**

**Opis:** Klasa **Zwierze** reprezentuje zwierzę. Każde zwierzę ma przypisane identyfikator, imię, gatunek, wiek oraz dane właściciela. Klasa implementuje interfejsy **ICloneable**, **IComparable<Zwierze>** i **IEquatable<Zwierze>**.

**Właściwości:**

1. **Imie**: Imię zwierzęcia (dostęp do odczytu i zapisu).
2. **Gatunek**: Gatunek zwierzęcia (dostęp do odczytu i zapisu).
3. **Wiek**: Wiek zwierzęcia (dostęp do odczytu i zapisu).
4. **ImieWlasciciela**: Imię właściciela zwierzęcia (dostęp do odczytu i zapisu).
5. **NazwiskoWlasciciela**: Nazwisko właściciela zwierzęcia (dostęp do odczytu i zapisu).
6. **TelefonKontaktowy**: Numer telefonu właściciela zwierzęcia (dostęp do odczytu i zapisu).
7. **Identyfikator**: Unikalny identyfikator zwierzęcia (dostęp do odczytu i zapisu).
8. **Licznik**: Licznik tworzenia nowych zwierząt (dostęp do odczytu i zapisu).

**Kontruktory**:

1. 1. **Zwierze()**: Konstruktor domyślny, inicjalizuje obiekt Zwierze nadając mu unikalny identyfikator.
2. **Zwierze(imieWlasciciela, nazwiskoWlasciciela, telefonKontaktowy, gatunek)**: Konstruktor parametryczny, ustawiający dane właściciela i gatunek zwierzęcia.
3. **Zwierze(imie, imieWlasciciela, nazwiskoWlasciciela, telefonKontaktowy, gatunek, wiek)**: Konstruktor parametryczny, ustawiający pełne informacje o zwierzęciu.

**Metody:**

1. **Equals(other)**: Sprawdza, czy dwa zwierzęta są identyczne na podstawie identyfikatora i imienia.
2. **Equals(obj)**: Przesłonięta metoda zwracająca wartość logiczną, czy obiekt jest równy danemu zwierzęciu.
3. **Zwierze()**: Konstruktor domyślny, inicjalizuje obiekt Zwierze nadając mu unikalny identyfikator.
4. **Zwierze(imieWlasciciela, nazwiskoWlasciciela, telefonKontaktowy, gatunek)**: Konstruktor parametryczny, ustawiający dane właściciela i gatunek zwierzęcia.
5. **Zwierze(imie, imieWlasciciela, nazwiskoWlasciciela, telefonKontaktowy, gatunek, wiek)**: Konstruktor parametryczny, ustawiający pełne informacje o zwierzęciu.
6. **WypiszZwierze()**: Zwraca tekstową reprezentację zwierzęcia w formie opisu.
7. **Clone()**: Implementacja interfejsu **ICloneable**, umożliwia klonowanie obiektu zwierzęcia.
8. **CompareTo(other)**: Implementacja interfejsu **IComparable<Zwierze>**, umożliwia porównywanie zwierząt na podstawie imienia.
9. **ToString()**: Przesłonięta metoda zwracająca tekstową reprezentację zwierzęcia.

**Klasa Wizyta**

**Opis:** Klasa **Wizyta** reprezentuje wizytę. Każda wizyta ma przypisane unikalne ID, lekarza, pacjenta (zwierzę), datę oraz powód wizyty. Klasa implementuje interfejs **IComparable<Wizyta>**.

**Właściwości:**

1. **Id\_wizyty**: Unikalny identyfikator wizyty.
2. **Lekarz**: Lekarz prowadzący wizytę.
3. **Zwierze**: Zwierzę będące pacjentem na wizycie.
4. **Data\_wizyty**: Data planowanej wizyty.
5. **Licznik\_wizyt**: Licznik tworzenia nowych wizyt.
6. **Powod**: Powód wizyty.

**Konstruktory:**

1. **Wizyta()**: Konstruktor domyślny, inicjalizuje obiekt Wizyta nadając mu unikalne ID.
2. **Wizyta(lekarz, zwierze, data\_wizyty, powod)**: Konstruktor parametryczny, ustawiający dane o lekarzu, pacjencie, dacie i powodzie wizyty.

**Metody**:

1. **WypiszWizyte()**: Zwraca tekstową reprezentację wizyty w formie opisu.
2. **CompareTo(other)**: Implementacja interfejsu **IComparable<Wizyta>**, umożliwia porównywanie wizyt na podstawie daty.
3. **ToString()**: Przesłonięta metoda zwracająca tekstową reprezentację wizyty.

**Klasa Klinika**

**Opis:** Klasa **Klinika** reprezentuje weterynaryjną klinikę. Klinika zarządza lekarzami, zwierzętami oraz wizytami. Implementuje interfejs **IZapisywalna** umożliwiający zapis i odczyt danych w formacie XML.

**Właściwości(dostęp do odczytu i zapisu):**

1. **Nazwa**: Nazwa kliniki.
2. **Lekarze**: Lista lekarzy pracujących w klinice.
3. **Zwierzeta**: Lista zwierząt będących pacjentami w klinice.
4. **Wizyty**: Lista planowanych wizyt w klinice.

**Konstruktory:**

1. **Klinika()**: Konstruktor domyślny, inicjalizuje obiekt Klinika.
2. **Klinika(nazwa)**: Konstruktor parametrowy, ustawiający nazwę kliniki.
3. **Klinika(nazwa, lekarze, zwierzeta, wizyty)**: Konstruktor parametrowy, ustawiający pełne dane o klinice.

**Metody:**

1. **DodawanieLekarza(l)**: Dodaje lekarza do listy lekarzy kliniki.
2. **DodawanieZwierzecia(z)**: Dodaje zwierzę do listy zwierząt kliniki.
3. **DodawanieWizyty(wizyta)**: Dodaje wizytę do listy wizyt kliniki, aktualizując harmonogram lekarza.
4. **SortujLekarzy()**: Sortuje listę lekarzy alfabetycznie.
5. **SortujZwierzeta()**: Sortuje listę zwierząt alfabetycznie.
6. **SortujWizyty()**: Sortuje listę wizyt według daty.
7. **UsuwanieLekarza(lekarz)**: Usuwa lekarza z listy lekarzy kliniki.
8. **UsuwanieZwierzecia(zwierze)**: Usuwa zwierzę z listy zwierząt kliniki.
9. **WypiszZwierzeta()**: Zwraca tekstową reprezentację wszystkich zwierząt w klinice.
10. **WypiszWizyty()**: Zwraca tekstową reprezentację wszystkich wizyt w klinice.
11. **WypiszZaplanowaneWizytyZwierzecia(zwierze)**: Zwraca tekstową reprezentację zaplanowanych wizyt dla danego zwierzęcia.
12. **WypiszOdbyteWizytyZwierzecia(zwierze)**: Zwraca tekstową reprezentację odbytych wizyt dla danego zwierzęcia.
13. **ToString()**: Przesłonięta metoda zwracająca tekstową reprezentację kliniki.
14. **ZapiszXML(nazwa)**: Implementuje interfejs **IZapisywalna**, zapisuje dane kliniki do pliku XML.
15. **OdczytXml(nazwa)**: Odczytuje dane kliniki z pliku XML