

Przychodnia

1(2pkt). Stwórz abstrakcyjną klasę `Osoba`. W tej klasie wykonaj następujące czynności:

- a) dodaj pole `imieNazwisko` typu `string` z mod. `protected`
- b) dodaj konstruktor z parametrem typu `string`, pobrany parametr należy ustawić jako pole `imieNazwisko`.

2(2pkt). Stwórz klasę `Pacjent` dziedziczącą z klasy `Osoba`. W klasie potomnej wykonaj następujące czynności:

- a) dodaj prywatne pola `wiek` (typu `int`) i `choroba` (typu `string`)
- b) dodaj konstruktor z trzema parametrami kolejno typów (`string`, `int`, `string`), pobrane parametry należy ustawić jako odpowiednia pola: `imieNazwisko`, `wiek`, `choroba`;
- c) przesłoń metodę `ToString()` tak, aby zwracała informacje o Pacjencie np. `Pacjent, imię i nazwisko: Jan Kowalski, wiek 50, choroba: grypa.`

3(2pkt). Stwórz klasę `Lekarz` dziedziczącą z klasy `Osoba`. W klasie potomnej wykonaj następujące czynności:

- a) dodaj prywatne pole `specjalnosc` typu `string`,
- b) dodaj konstruktor z dwoma parametrami typu `string`, pobrane parametry należy ustawić kolejno jako pola `imieNazwisko`, `specjalnosc`
- c) przesłoń metodę `ToString()` tak, aby zwracała informacje o Lekarzu np.: `Lekarz, imię i nazwisko: Anna Nowak, specjalność: internista.`

4(2pkt). Stwórz interfejs `IPrzychodnia` i dodaj w nim deklarację następujących metod:

`WykonajPorade()` – typ zwracany `string`
`WykonajBadanie()` – typ zwracany `string`

5(2pkt) Podepnij interfejs `IPrzychodnia` do klas `Lekarz` i `Pacjent`. Dodaj implementację metod w taki sposób aby zwracały odpowiednio `string` i informacją o badaniu lub poradzie i nazwie klasy np.

- „Wykonano poradę! – Lekarz”
- „Wykonano badanie! – Pacjent”

6(2 pkt) W klasie Program i instrukcji Main stwórz program testujący następująco:

- a) stwórz generyczną kolejkę przechowującą obiekty typu Pacjent (nazwij ją pacjenci)
- b) do kolejki dodaj co najmniej 5 obiektów za pomocą konstruktora parametrycznego
- c) z użyciem metody dostępnej tylko dla kolejek usuń z niej pierwszy element oraz jednocześnie napisz na konsoli dane o tym pacjencie, potem zwróć string uzyskany przez metodę WykonajPorade() dla obiektu klasy Pacjent
- d) wypisz na konsoli wszystkie elementy z kolejki na ten moment

7(2pkt) Do klasy Lekarz podepnij interfejs IComparable<T> i zaimplementuj metodę CompareTo() tak by sortowanie odbywało się po specjalności (odwrotnie alfabetycznie od Z do A)

8(2 pkt) W klasie Program i instrukcji Main wykonaj czynności:

- a) zrób linijkę komentarza po kodzie z polecenia 6
- b) stwórz listę o nazwie przychodnia na obiekty typu Lekarz
- c) dodaj na nią 5 obiektów typu Lekarz za pomocą konstruktora parametrycznego (co najmniej dwa muszą mieć tę samą specjalność)
- d) posortuj listę i po sortowaniu wypisz elementy na konsoli

9(2pkt) Wykonaj w całym projekcie następujące czynności:

- a) wyrzuć „ręcznie” wyjątek w sytuacji gdy wiek pacjenta dodawanego na listę pacjenci będzie liczbą ujemną
- b) za pomocą bloku try...catch obsłuż wyjątek – na konsoli ma pojawić się komunikat dla użytkownika np. Wiek nie może być liczbą ujemną.
- c) dodaj również ograniczenie górne, aby wiek był mniejszy niż 120

10(2pkt) Wykonaj czynności:

- a) Do klasy Pacjent podepnij interfejs ICloneable i zaimplementuj mechanizm zwany płytką kopią.
- b) Stwórz generyczny stos o nazwie pacjenci2 na obiekty typu Pacjent.

c) za pomocą instrukcji foreach skopiuj każdy element z kolejki pacjenci i umieść go w na stosie pacjenci2, po całej operacji kolejka pacjenci powinna zostać pusta

d) wypisz elementy stosu na konsoli

Max – 20 pkt.

bardzo dobry (5,0) - 91-100%;

dobry plus (4,5) - 86-90%;

dobry (4,0) - 75-85%;

dostateczny plus (3,5) - 70-74%;

dostateczny (3,0) - 50-69%;

niedostateczny (2,0) - 0-49%.