Wprowadzenie do programowania zorientowanego obiektowo

Modelowanie obiektowe

1. Wypożyczalnia pojazdów zajmuje się wypożyczaniem samochodów. Posiada w swojej ofercie zarówno pojazdy osobowe, jak i dostawcze.
2. Każdy z pojazdów jest opisany za pomocą jego marki, informacji, czy pojazd jest aktualnie dostępny do wypożyczenia, jaki jest numer rejestracyjny pojazdu, ile razy samochód był już wypożyczany oraz jaki posiada przebieg w km.
3. Możliwe jest utworzenie listy wszystkich pojazdów znajdujących się na stanie wypożyczalni oraz listy pojazdów, które w danej chwili dostępne są do wypożyczenia.

Środowisko zintegrowane BlueJ

4. Zapoznaj się ze środowiskiem zintegrowanym BlueJ
    a. Informacje wstępne
    b. http://www.bluej.org/

Klasy i obiekty

5. Utwórz nowy projekt o nazwie WypozyczalniaPojazdow
6. Utwórz klasę Auto
7. Dodaj pola
    a. String marka
    b. boolean wypozyczone
8. Dodaj konstruktor

```
public Auto(String marka, boolean wypozyczone){
    this.marka = marka;
    this.wypozyczone = wypozyczone;
}
```

9. Dodaj kolejny konstruktor `Auto(String marka)`
10. Utwórz kilka obiektów (instancji klasy Auto) wykorzystując pierwszy z konstruktorów. Dokonaj inspekcji obiektów (wartości pól obiektów). Jakie wartości posiadają te pola?
11. Utwórz kilka obiektów (instancji klasy Auto) wykorzystując drugi z konstruktorów. Dokonaj inspekcji obiektów (wartości pól obiektów). Jaką wartość posiada pole „wypożyczone”? Dlaczego to pole posiada wartość pomimo, iż nie została ona przekazana w momencie wywołania konstruktora?

Okno instrukcji Code Pad

12. Załącz Code Pad i zapoznaj się ze sposobem wprowadzania instrukcji, jedno i wielowierszowych. Aby wprowadzić kolejną linię instrukcji naciśnij Shift+Enter.
13. Wyświetl swoje imię i nazwisko.
14. Wyświetl obliczoną sumę dwóch dowolnych liczb naturalnych.
15. Oblicz i wyświetl sumę liczb naturalnych z przedziału <1,20>; wykorzystaj instrukcję for.
16. Korzystając z instrukcji `new Auto (…)` utwórz nowy obiekt i umieść go w oknie obiektów (przeciągnij go do okna obiektów).

Dodawanie i wywoływanie metod

17. Dodaj do klasy Auto metodę uruchomKlakson() wyświetlającą sygnał klaksonu „Beep beep”. Wykorzystaj instrukcję System.out.println(…);
18. Utwórz obiekt i wywołaj metodę uruchomKlakson()
19. Dodaj do klasy Auto metodę toString(), zwracającą markę pojazdu oraz informację, czy auto jest wypożyczone, czy dostępne

```
public String toString(){

    String dostepnosc = wypozyczone ? "wypozyczone" : "dostępne";

    return marka + ", " + dostepnosc;

}
```

20. Utwórz kilka obiektów klasy Auto.
21. Wywołaj metodę toString(), wybierając ją z listy dostępnych metod. Sprawdź w inspektorze obiektów, jaką wartość zwraca wywołana.
22. Wyświetl informacje o pojeździe - wywołaj metodę toString() wprowadzając ją w oknie instrukcji (Code Pad), każdym ze sposobów:
    a. System.out.println(nazwaObiektu.toString());
    b. System.out.println(nazwaObiektu);

    Dlaczego w tym drugim przypadku nie było konieczne podanie nazwy metody, a jedynie nazwy obiektu?

23. Korzystając z okna instrukcji (Code Pad), utwórz obiekt z jednoczesnym wyświetleniem informacji o pojeździe: `System.out.println(new Auto(„Fiat”));`

Wyrażanie związków (relacji) pomiędzy klasami – kompozycja

24. Utwórz klasę Wypozyczalnia.
25. Dodaj pole reprezentujące listę pojazdów, jakie posiada wypożyczalnia (dodaj również instrukcję `import`, aby określić, gdzie znajduje się klasa ArrayList):
    `ArrayList<Auto> pojazdy = new ArrayList<Auto>();`
26. W klasie Wypozyczalnia dodaj metodę, która umożliwi dodanie kolejnego pojazdu do wypożyczalni:

```
/**
```

```
 * Dodaj pojazd
 */
public void dodajPojazd(Auto a){
    pojazdy.add(a);
}
```

27. Zwróć uwagę jak zmienił się schemat klas oraz oznaczenie relacji zachodzącej pomiędzy dwoma klasami.
28. Utwórz 3 pojazdy - obiekty klasy Auto.
29. Utwórz obiekt Wypozyczalnia.
30. Sprawdź w inspektorze obiektów, ile pojazdów znajduje się w wypożyczalni.
31. Dodaj utworzone 3 pojazdy do wypożyczalni. Wykorzystaj metodę dodajPojazd()
32. Sprawdź w inspektorze obiektów, ile pojazdów znajduje się w wypożyczalni.
33. W klasie Wypozyczalnia dodaj metodę listaPojazdow() wyświetlającą wykaz pojazdów znajdujących się w wypożyczalni (wykaz obiektów znajdujących się w tablicy). Wykorzystaj instrukcję `for`. Sprawdź w API Java wykaz metod klasy ArrayList. Odszukaj metodę:
    a. zwracającą liczbę elementów tablicy
    b. zwracającą n-ty obiekt tablicy

    Wykorzystaj te metody w instrukcji `for`.
34. Utwórz wypożyczalnię, dodaj 3 pojazdy, a następnie wyświetl listę pojazdów.

Wyrażanie związków (relacji) pomiędzy klasami – dziedziczenie

35. Utwórz klasę AutoOsobowe, będącą klasą pochodną klasy Auto:

```
public class AutoOsobowe extends Auto
   {
       int miejsca; // liczba miejsc w pojeździe
       /**
        * Konstruktor
        */
       public AutoOsobowe(String marka, boolean wypozyczone, int
   miejsca){
           super(marka, wypozyczone); // wywołanie konstruktora nadklasy
           this.miejsca = miejsca;
       }
   }
```

Zwróć uwagę na potrzebę wywołania konstruktora klasy dziedziczonej (nadklasy).

36. Utwórz klasę AutoDostawcze, będącą klasą pochodną klasy Auto.
    a. Dodaj pole `double ladownosc` oznaczające ładowność pojazdu w tonach.
    b. Nie zapomnij o wywołaniu konstruktora klasy nadrzędnej.
37. Zwróć uwagę na schemat klas oraz oznaczenie relacji zachodzącej pomiędzy klasami dziedziczącymi.
38. Utwórz jedno auto osobowe i jedno dostawcze. Sprawdź w inspektorze obiektów wartości pól.
39. Utwórz wypożyczalnię i dodaj obydwa utworzone auta. Zwróć uwagę na typ obiektów, które dodajesz oraz typ klasy ArrayList. Czy dostrzegasz różnicę? Dlaczego taka operacja jest dozwolona?
40. Sprawdź w inspektorze obiektów, ile pojazdów posiada wypożyczalnia.
41. Wyświetl listę pojazdów.
42. Dodaj w klasie AutoOsobowe metodę toString() zwracającą markę, dostępność oraz liczbę miejsc w pojeździe.
43. Dodaj w klasie AutoDostawcze metodę toString() zwracającą markę, dostępność oraz ładowność pojazdu.
44. Utwórz obiekty i wyświetl listę pojazdów dostępnych w wypożyczalni. Porównaj metody toString() klasy Auto, AutoOsobowe i AutoDostawcze. Które z nich są wywoływane i dlaczego?

Klasa abstrakcyjna

45. Zmień klasę Auto na abstrakcyjną.
46. Czym charakteryzuje się klasa abstrakcyjna?
47. Dodaj obiekty i wyświetl listę pojazdów wypożyczalni.
48. Dodaj metodę abstrakcyjną String wymaganyDokument(). Następnie zaimplementuj tą metodę w klasach pochodnych. Dla auta osobowego metoda powinna zwrócić tekst „wymagane prawo jazdy jat. B”, a dla auta dostawczego „wymagane prawo jazdy kat. C”.
49. Zmodyfikuj metody toString() w klasach pochodnych, aby wyświetlały również informację o wymaganym dokumencie. Wykorzystaj metodę wymaganyDokument()
50. Wyświetl listę pojazdów wypożyczalni.

Metoda main()

51. Dodaj w klasie Wypozyczalnia metodę main().
52. W metodzie main() umieść instrukcje:
    a. Tworzące dwa obiekty klasy AutoOsobowe
    b. Tworzące dwa obiekty klasy AutoDostawcze.
    c. Tworzące obiekt Wypozyczalnia
    d. Dodające auta do wypożyczalni.
    e. Wyświetlające listę pojazdów wypożyczalni.
53. Wywołaj metodę main() wybierając ją z listy dostępnych metod w klasie Wypozyczalnia

Interface

54. Utwórz interfejs o nazwie Dostepnosc.
55. Dodaj w interfejsie nagłówek metody: String listaPojazdowDostepnych(); która wyświetlać będzie listę dostępnych pojazdów w wypożyczalni (aktualne nie wypożyczonych).

56. Zaimplementuj interfejs w klasie Wypozyczalnia. Czy możliwe jest teraz skompilowanie klasy?
57. Dodaj metodę listaPojazdowDostepnych() w klasie Wypozyczalnia. Wprowadź w metodzie zbiór instrukcji, które wyświetlą tylko te pojazdy, które nie są wypożyczone.
58. Sprawdź działanie dodanej metody.

Dodatkowe modyfikacje aplikacji

59. Zmodyfikuj wyświetlanie listy pojazdów, aby była ona numerowana. Wyświetl ponumerowaną listę pojazdów.
60. Każdy pojazd posiada numer rejestracyjny. Uzupełnij klasę Auto o pole String nrRejestracyjny. Wprowadź takie zmiany w programie, aby wyświetlana lista pojazdów zawierała również ich numer rejestracyjny. Wyświetl listę pojazdów.
61. Każdy z pojazdów posiada przebieg w km.
    a. Dodaj tablicę o nazwie „przebieg”, w której będziesz ewidencjonował przebiegi pojazdu w km osiągnięte w kolejnych wypożyczeniach. W której klasie należy ją dodać?
    b. Czy tablica ta powinna mieć z góry określoną liczbę komórek?
    c. Jaki kontener należy zastosować?
    d. Dodaj metodę dodajPrzebieg(int) umożliwiającą dodanie kolejnego przebiegu samochodu (liczba km przejechanych podczas ostatniego wypożyczenia).
    e. Dodaj metodę int razemPrzebieg() obliczającą łączną liczbę przejechanych km przez samochód.
    f. Wprowadź takie zmiany w programie, aby wyświetlana lista pojazdów zawierała również informacje o ich przebiegu. Wyświetl listę pojazdów.
62. Wypożyczalnia powinna również posiadać informację o liczbie wypożyczeń samochodu.
    a. Dodaj metodę int liczbaWypozyczen(), która obliczy, ile razy samochód był wypożyczany. Aby to obliczyć, odczytaj liczbę elementów tablicy „przebieg”.
    b. Wprowadź takie zmiany w programie, aby wyświetlana lista pojazdów zawierała również informacje o liczbie ich wypożyczeń. Wyświetl listę pojazdów.

Diagram klas