Îndrumător laborator – LUCRAREA NR. 6

Clase și obiecte

# Exercițiu rezolvat

Creați o clasă Java numită CarInsurancePolicy care conține trei câmpuri care rețin numărul poliței de asigurare, numărul de plăți pe care titularul poliței le va efectua anual și orașul de reședință al asiguratului. Creați un constructor care necesită parametri pentru toate cele trei câmpuri de date.

Să presupunem că agenția care vinde polițe de asigurare auto se află în orașul Mayfield. Creați un constructor cu doi parametri, numărul poliței de asigurare și numărul de plăți. Acest constructor atribuie Mayfield membrului residentCity. Adăugați un al treilea constructor care necesită doar un parametru, numărul poliței de asigurare. Acest constructor folosește valorile implicite ale două plăți anuale și Mayfield ca oraș rezident. Adăugați o metodă de afișare care afișează toate datele poliței de asigurare. Salvați fișierul ca CarInsurancePolicy.java.

Creați o aplicație scurtă care să demonstreze utilizarea constructorilor. Aplicația declară trei obiecte CarInsurancePolicy folosind de fiecare dată o versiune de constructor diferită. Afișați datele pentru fiecare obiect. Salvați fișierul ca Createpolicies.java.

public class CarInsurancePolicy

{

private int policyNumber;

private int numPayments;

private String residentCity;

public CarInsurancePolicy(int num, int payments, String city)

{

policyNumber = num;

numPayments = payments;

residentCity = city;

}

public CarInsurancePolicy(int num, int payments)

{

⬄ this(num, payments, "Mayfield");

policyNumber = num;

numPayments = payments;

residentCity = "Mayfield";

}

public CarInsurancePolicy(int num)

{

⬄ this(num, 2);

policyNumber = num;

numPayments = 2;

residentCity = "Mayfield";

}

public void display()

{

System.out.println("Policy #" + policyNumber + ". " +

numPayments + " payments annually. Driver resides in " +

residentCity + ".");

}

}

public class CreatePolicies

{

public static void main(String[] args)

{

CarInsurancePolicy first = new CarInsurancePolicy(123);

CarInsurancePolicy second = new CarInsurancePolicy(456, 4);

CarInsurancePolicy third = new CarInsurancePolicy(789, 12, "Newcastle");

first.display();

second.display();

third.display();

}

}

# Exerciții propuse

1. Zarurile sunt utilizate în multe jocuri. Un zar poate fi aruncat pentru a arăta la întâmplare o valoare de la 1 la 6. Proiectați o clasă Die care poate reține un câmp de date întreg pentru o valoare (de la 1 la 6). Includeți un constructor care atribuie aleatoriu o valoare unui obiect zar. Pentru a înțelege complet procesul de generare a numerelor aleatorii, trebuie să aflați mai multe despre clasele și metodele Java. Cu toate acestea, deocamdată, puteți copia următoarea instrucțiune pentru a genera un număr aleator între 1 și 6 și a o atribui unei variabile. Utilizarea acestei afirmații presupune că ai atribuit valori adecvate constantelor statice.

randomValue 5 ((int) (Math.random () \* 100)% HIGHEST\_DIE\_VALUE + LOWEST\_DIE\_VALUE);

De asemenea, includeți o metodă în clasa Die pentru a returna valoarea unui zar. Salvați clasa ca Die.java.

Acum folosiți clasa pentru a crea un joc de zar simplu în care utilizatorul alege un număr între 2 (cel mai mic total posibil din două zaruri) și 12 (cel mai mare total posibil). Utilizatorul „aruncă” două zaruri de până la trei ori. Dacă numărul ales de utilizator apare, acesta câștigă și jocul se încheie. Dacă numărul nu apare în trei aruncări, computerul câștigă.Salvați aplicația ca TwoDice.java.

1. Carly’s Catering oferă mese pentru petreceri și evenimente speciale. În lucrarea 5, ați creat o clasă pentru un eveniment. Clasa Event conține două câmpuri public static final care rețin prețul per oaspete (35 USD) și valoarea limită pentru un eveniment mare (50 de invitați) și trei câmpuri private care rețin numărul evenimentului, numărul de invitați pentru eveniment și prețul. De asemenea, conține două metode publice set și trei metode publice get.

Acum, modificați clasa Event pentru a conține doi constructori supraîncărcați.

* Un constructor acceptă ca parametri numărul evenimentului și numărul de invitați. Transmiteți aceste valori metodelor setEventNumber() și, respectiv, setGuests(). Metoda setGuests() va calcula automat prețul evenimentului.
* Celălalt constructor este un constructor implicit care transmite „A000” și 0 la constructorul cu doi parametri.
* În prezent, clasa conține un câmp care reține prețul unui eveniment. Acum adăugați un alt câmp care reține prețul per oaspete și adăugați o metodă publică pentru a-i returna valoarea.
* În prezent, clasa conține o constantă pentru prețul per oaspete. Înlocuiți câmpul cu două câmpuri - un preț mai mic per oaspete care este de 32 USD și un preț mai mare per oaspete care este de 35 USD.
* Adăugați o nouă metodă numită isLargeEvent() care să returneze adevărat dacă numărul de invitați este de 50 sau mai mare iar în caz contrar returnează fals.
* Modificați metoda care stabilește numărul de invitați, astfel încât un eveniment mare (mai mult de 50 de invitați) să utilizeze prețul mai mic per oaspete pentru a seta noul câmp pricePerGuest și pentru a calcula prețul total al evenimentului. Un eveniment mic folosește prețul mai mare.

Salvați fișierul ca Event.java.

1. În lucrarea 4, ați creat, de asemenea, o clasă EventDemo pentru a demonstra utilizarea clasei Event creând două obiecte. Acum, modificați acea clasă pentru a crea două obiecte eveniment și includeți următoarele metode noi în clasă:

* Instanțiați un obiect pentru a păstra valorile implicite ale constructorului.
* Acceptați date introduse de utilizator pentru numărul evenimentului și câmpurile de invitați și utilizați acest set de date pentru a instanția cel de-al doilea obiect.
* Afișați toate detaliile pentru ambele obiecte.
* Înstanțiați un obiecte eveniment și solicitați utilizatorului valori pentru fiecare obiect.
* Modificați metoda care afișează detaliile evenimentului pentru a utiliza noua metodă isLargeEvent() și noul preț pe valoare de invitat. Utilizați metoda de afișare pentru obiect.
* Creați o metodă care acceptă două obiecte de eveniment și o returnează pe cea mai mare în funcție de numărul de invitați. (Dacă evenimentele au același număr de invitați, puteți returna oricare obiect.) Apelați această metodă de trei ori - o dată cu fiecare pereche de evenimente instanțiate - și afișați numărul evenimentului și numărul de invitați pentru fiecare argument, precum și numărul evenimentului. și numărul de invitați pentru evenimentul mai mare.

Salvați fișierul ca EventDemo.java.

1. Modificați clasa EventDemo astfel încât:

* Solicitați numărul de invitați pentru fiecare eveniment până când valoarea este între 5 și 100 inclusiv.
* Pentru unul dintre obiectele Event, creați o buclă care să afișeze *Please come to my event!* pentru fiecare invitat.

1. Sammy’s Seashore Supplies închiriază echipamente pe plajă, precum caiace, canoe, scaune de plajă și umbrele pentru turiști. În lucrarea 5, ați creat o clasă de închiriere pentru companie. Clasa Rental conține două câmpuri statice publice finale care conțin numărul de minute într-o oră și rata de închiriere pe oră (40 USD) și patru câmpuri private care rețin un numărul de contract, numărul de ore pentru închiriere, numărul de minute peste o oră, și prețul. De asemenea, conține două metode publice set și patru metode publice get.

Acum, modificați clasa Rental pentru a conține doi constructori supraîncărcați.

* Un constructor acceptă un număr de contract și un număr de minute ca parametri. Treceți aceste valori la metodele setContractNumber() și, respectiv, setHoursAndMinutes(). Metoda setHoursAndMinutes() va calcula automat orele, minutele suplimentare și prețul.
* Celălalt constructor este un constructor implicit care transmite „A000” și 0 la constructorul cu doi parametri.
* În prezent, prețul de închiriere este calculat la 40 USD pe oră, plus 1 dolar pentru fiecare minut care depășește o oră întreagă. Acest lucru înseamnă că un client care închiriază echipament timp de 41 sau mai multe minute peste o oră plătește mai mult decât un client care așteaptă până la următoarea oră pentru a returna echipamentul. Modificați calculul prețului astfel încât un client să plătească 40 USD pentru fiecare oră completă și 1 dolar pentru fiecare minut suplimentar până la 40 de minute inclusiv.

Salvați fișierul ca Rental.java.

1. În lucrarea 4, ați creat, de asemenea, o clasă RentalDemo pentru a demonstra utilizarea unui obiect Rental. Acum, modificați acea clasă pentru a instanția două obiecte Rental.

* Instanțiați un obiect pentru a păstra valorile implicite ale constructorului.
* Acceptați datele furnizate de utilizator pentru câmpurile număr și minute ale contractului și utilizați acest set de date pentru a instanția cel de-al doilea obiect. Afișați toate detaliile pentru ambele obiecte.
* Instanțiați încă un obiect și solicită utilizatorului valori pentru fiecare caracteristică. Afișați detaliile pentru obiect pentru a verifica dacă noul calcul al prețului funcționează corect.
* Creați o metodă care acceptă două obiecte de închiriere și returnează-l cu cel mai lung timp de închiriere. (Dacă închirierile au același timp, puteți returna oricare obiect.) Apelați această metodă de trei ori - o dată cu fiecare pereche de închirieri instantanee - și afișați numărul și ora contractului în ore și minute pentru fiecare argument, precum și numărul contractului cu cea mai lungă durată de închiriere.

Salvați fișierul ca RentalDemo.java.

1. Modificați clasa RentalDemo pentru a face următoarele:

* Citiți continuu numărul de minute al fiecărei închirieri până când valoarea este între 60 și 7.200 inclusiv.
* Pentru unul dintre obiectele de închiriat, creați o buclă care afișează un cupon *Coupon good for 10 percent off next rental* de câte ori există ore întregi de închiriere.

Salvați fișierul modificat ca RentalDemo.java.