Beskrivning av kod och funktioner

1. Basklassen: Vehicle

Basklassen `Vehicle` är en abstrakt klass som definierar gemensamma attribut och metoder för alla fordon. Den representerar kärnan i projektets design och hjälper till att återanvända kod genom att centralisera gemensamma funktioner.

Attribut:

- `model` (fordonsmodell)
- `registrationNumber` (registreringsnummer)
- `dailyRate` (hyreskostnad per dag)
- `rented` (indikerar om fordonet är uthyrt)
- `rentalDays` (antal dagar fordonet är uthyrt)

Metoder:

- `getVehicleType()` är abstrakt och implementeras i subklasser för att specificera fordonstyper.
- `rent()` hanterar uthyrningslogik, inklusive att ange antal hyresdagar.
- `returnVehicle()` återställer uthyrningsstatus.
- `calculateRentalCost()` beräknar totalkostnaden för uthyrningen.

Denna abstrakta klass fungerar som en mall och bidrar till att minska redundans genom att samla gemensamma egenskaper på en plats.

2. Specifika Fordonsklasser

Fyra subklasser (`Car`, `SUV`, `Truck`, `Motorcycle`) ärver från `Vehicle` och lägger till unika egenskaper:

- **`Car`:** Har en flagga för luftkonditionering.
- **`SUV`:** Inkluderar en egenskap för fyrhjulsdrift.
- **`Truck`:** Har ett attribut för lastkapacitet.
- **`Motorcycle`:** Kan ha en sidovagn.

Dessa klasser visar arvets styrka genom att återanvända logik från basklassen samtidigt som de kan lägga till specialiserade funktioner.

3. Interfacet: Rentable

Interfacet `Rentable` definierar ett kontrakt för uthyrning. Det används av `Vehicle` och därmed indirekt av alla dess subklasser.

Metoder:

- `rent(int days)`: Startar en uthyrning.
- `returnVehicle()`: Markerar ett fordon som återlämnat.
- `isRented()`: Returnerar uthyrningsstatus.
- `getRentalCost()`: Hämtar total hyreskostnad.
- `calculateRentalCost(int days)`: Beräknar kostnad baserat på antal dagar.

Genom att använda ett interface separeras funktionalitet från implementation. Detta gör systemet mer flexibelt och enkelt att lägga till / utöka vid önskemål.

4. Konsol- Interaktion

Applikationen har ett användargränssnitt där användare kan:

1. Välja ett fordon från en lista.
2. Ange antal hyresdagar.
3. Hyra fordonet och se kostnaden.

Gränssnittet är intuitivt och gör det enkelt för användaren att navigera genom applikationen.

Objektorienterade Principer i Projektet

1. **Abstraktion:** Abstraktion används genom basklassen `Vehicle`, som döljer implementeringsdetaljer och endast exponerar nödvändiga attribut och metoder. Detta gör det enkelt för utvecklare att använda klassen utan att förstå dess interna logik.
2. **Inkapsling:** Alla attribut i klasserna är privata och åtkomst till dessa sker via getter- och setter-metoder. Detta skyddar datan från direkt manipulation och bevarar objektets integritet.
   Exempel:
   - `getDailyRate()` ger åtkomst till daglig hyreskostnad utan att direkt exponera attributet.
3. **Arv:** Arv demonstreras genom att subklasserna (`Car`, `SUV`, `Truck`, `Motorcycle`) ärver från `Vehicle`. Detta möjliggör återanvändning av gemensamma attribut och metoder samt tillåter specialisering där det behövs.
4. **Polymorfism:** Polymorfism används via metoden `getVehicleType()` som implementeras olika av varje subklass. Vilket innebär att samma metodnamn kan utföra olika uppgifter beroende på objektets typ.