RELAZIONE PROGETTO DI MODELLAZIONE E GESTIONE DELLA CONOSCENZA

DAMIANO MONTANARO (122407)

SOMMARIO

1. DESCRIZIONE DEL DOMINIO DELL'APPLICAZIONE

L'applicazione è un configuratore di auto che consente agli utenti di personalizzare vari aspetti di un'auto, tra cui il tipo, la marca, il prezzo, il colore, il motore, le ruote e le opzioni aggiuntive. Il dominio comprende diversi tipi di auto, come le auto di lusso, le auto sportive e le auto elettriche, ciascuna con caratteristiche e requisiti specifici. Il configuratore mira a fornire un'interfaccia facile da usare per la selezione e la configurazione dei componenti dell'auto, convalidando gli input dell'utente.

2. DESCRIZIONE DEI CONCETTI E DELLE PROPRIETÀ MODELLATE NELL'ONTOLOGIA

L'ontologia modella i seguenti concetti e proprietà:

1) CLASSI:

-**Car**: Rappresenta un'auto generica configurabile.

-**ElectricCar**: Una sottoclasse di Car, che rappresenta le auto elettriche.

-**LuxuryCar**: Una sottoclasse di Car, che rappresenta le auto lussuose.

-**SportsCar**: Una sottoclasse di Car, che rappresenta le auto sportive.

-**Engine**: Rappresenta il motore di un’auto.
-**Wheel**: Rappresenta la ruota di un’auto.
-**Component**: Rappresenta un componente di un’auto.
-**Option**: Rappresenta un'opzione selezionabile per un’auto o un componente.
-**Material**: Una sottoclasse di Option, che rappresenta le opzioni di materiale per i componenti.
-**Color**: Sottoclasse di Option, che rappresenta le opzioni di colore per le auto.
-**Company**: Rappresenta un'azienda che produce auto.

2) PROPRIETÀ:
-**Brand**: La marca di un'auto (data property)
-**Color**: Il colore dell'auto (data property)
-**Price**: Il prezzo di un’auto (data property)
-**horsePower**: I cavalli di un motore (data property)
-**weight**: Il peso di un componente dell’auto (data property)
-**warrantyPeriod**: Il periodo di garanzia di un’auto (data property)
-**hasComponent**: Collega un’auto ai suoi componenti (object property)
-**hasOption**: Collega un componente alle sue azioni disponibili (object property)
-**compatibleWith**: Indica la compatibilità tra i componenti (object property)
-**producedBy**: Collega un’auto alla sua marca (object property)

3) SPARQL QUERY:
Sono state eseguite le seguenti SPARQL QUERY:

a) Ottieni tutte le macchine e le loro marche:

```
PREFIX ex: <http://studentproject.org/car-configurator#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?car ?brand
WHERE {
  ?car rdf:type ex:Car .
  ?car ex:brand ?brand .
}
```

b) Ottieni tutte le macchine elettriche e i loro prezzi:

```
PREFIX ex: <http://studentproject.org/car-configurator#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?electricCar ?price
WHERE {
  ?electricCar rdf:type ex:ElectricCar .
  ?electricCar ex:price ?price .
}
```

c) Ottieni tutte le macchine con i loro colori e prezzi:

```
PREFIX ex: <http://studentproject.org/car-configurator#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?car ?color ?price
WHERE {
```

```
  ?car rdf:type ex:Car .
  ?car ex:color ?color .
  ?car ex:price ?price .
}
```

3. DESCRIZIONE DELLE RESPONSABILITÀ NELLO SVILUPPO DELL'APPLICAZIONE

-**Interfaccia utente**: Gestisce l'input dell'utente e visualizza le opzioni di configurazione dell'auto. Assicura un'esperienza utente fluida e intuitiva.
-**Validazione**: Assicura che gli input dell'utente rientrino negli intervalli e nei formati specificati. Previene le configurazioni non valide.
-**Configurazione**: Gestisce la creazione e la personalizzazione degli oggetti dell'auto in base agli input dell'utente. Assicura che tutte le opzioni e i componenti selezionati siano compatibili.

-**Gestione degli errori**: Fornisce un feedback all'utente in caso di input non validi o di errori. Assicura che l'utente sia informato di eventuali problemi e possa correggerli.

4. DESCRIZIONE DELLE RESPONSABILITÀ DELLE CLASSI E DELLE INTERFACCE

CLASSI

LuxuryCar.java, SportsCar.java, ElectricCar.java: Sottoclassi di Car, ciascuna con proprietà e metodi specifici. Rappresentano diversi tipi di auto con caratteristiche uniche.

Engine.java: Rappresenta il motore di un'auto con proprietà come i cavalli. Fornisce metodi per configurare il motore.

Wheel.java: Rappresenta una ruota con proprietà come il peso. Fornisce metodi per configurare la ruota.

Material.java: Rappresenta le opzioni di materiale per i componenti.

BasicOption.java: Rappresenta un'opzione di base per la configurazione dell'auto.

CarImpl.java: Rappresenta un'implementazione standard dell'auto nel sistema di configurazione dell'auto. Implementa l'interfaccia Car.java.

Company.java: Rappresenta una compagnia/un brand con un nome.

OptionImpl.java: Rappresenta un'implementazione dell'interfaccia Option.java.

INTERFACCE

Car.java (interfaccia): Rappresenta un'auto generica con proprietà come marca, prezzo, colore, motore e ruote. Fornisce metodi per aggiungere componenti e opzioni.

Component.java (interfaccia): Rappresenta un componente nel sistema di configurazione dell'auto.

Optionable.java (interfaccia): Rappresenta un'entità a cui possono essere aggiunte delle opzioni.

Configurable.java (interfaccia): Rappresenta un componente configurabile nel sistema di configurazione dell'auto.

Option.java (interfaccia): Rappresenta un'opzione per la configurazione dell'auto.

CONTROLLER

CarConfigurator.java: Gestisce la creazione della macchina.

WheelsHandler.java: Gestisce l'input e la validazione del numero di ruote e del loro peso.

CreateCar.java: Crea la macchina secondo gli input dell'utente.

UserInterfaceController.java: Funge da controller per l'UI.

LE CLASSI PRINCIPI

UserInterface.java: Rappresenta l'UI (realizzata con un file fxml).

App.java: Fa partire l'applicazione con l'UI.

5. DESCRIZIONE DELL'USO DELL'APPLICAZIONE

L'applicazione consente agli utenti di configurare un'auto selezionandone il tipo, la marca, il prezzo, il colore, le specifiche del motore e le opzioni aggiuntive. L'interfaccia utente fornisce campi di input e scelte per ogni aspetto configurabile. L'applicazione convalida gli input e crea un oggetto auto in base alle selezioni dell'utente. L'applicazione fornisce un feedback in caso di errori.

Maggiori dettagli sul file README.md.

COME FAR PARTIRE IL PROGETTO?

Per prima cosa, bisogna estrarre i file dalla cartella .zip, poi eseguire, se si è su un terminale come Git Bash o Command Prompts, i percorsi `cd desktop` (per esempio, se la cartella è estratta lì) e `cd MontanaroDamiano122407`. Subito dopo, seguire i seguenti passaggi:

1. Vai alla directory di CarConfigurator: `cd CarConfigurator`
2. Scrivi il comando `gradle build`
3. Scrivi il comando `gradle run`

Così che parte l'applicazione con l'interfaccia grafica. Maggiori dettagli sul file README.md.