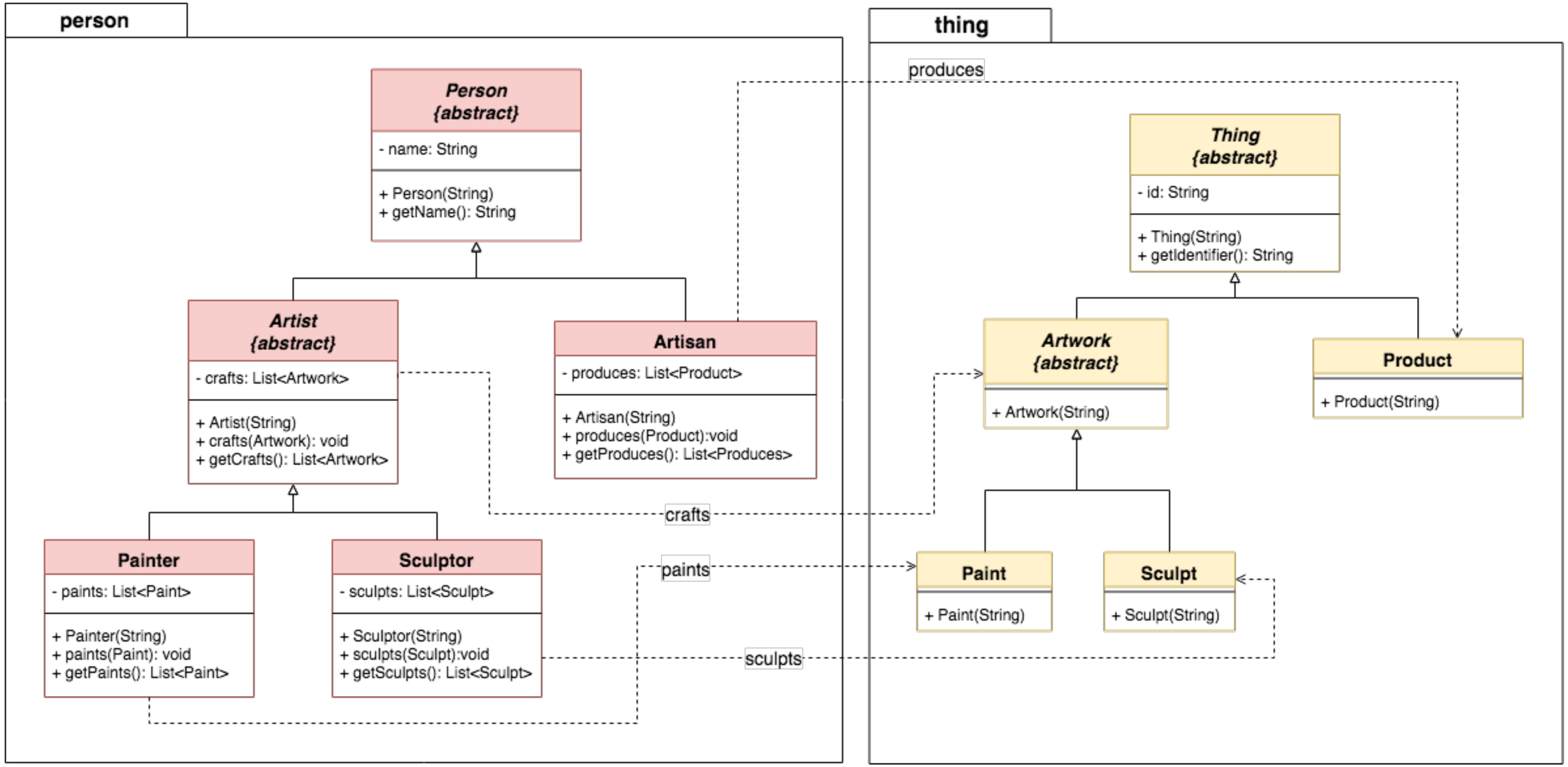
DEMO - ART MARKET



DEMO - ART MARKET

$$Artist \sqsubseteq Person$$

$$Artisan \sqsubseteq Person$$

$$Painter \sqsubseteq Artist$$

$$Sculptor \sqsubseteq Artist$$

$$Artwork \sqsubseteq Thing$$

$$Product \sqsubseteq Thing$$

$$Paint \sqsubseteq Artwork$$

$$Sculpt \sqsubseteq Artwork$$

$$paints \dot{\sqsubseteq} crafts$$

$$sculpts \dot{\sqsubseteq} crafts$$

$$Dom(crafts, Artist)$$

$$Rng(crafts, Artwork)$$

$$Dom(produces, Artisan)$$

$$Rng(produces, Product)$$

$$Dom(paints, Painter)$$

$$Rng(paints, Paint)$$

$$Dom(sculpts, Sculptor)$$

$$Rng(sculpts, Sculpt)$$

$$Rng(name, String)$$

$$Dom(name, Person)$$

$$Fun(name)$$

$$Rng(id, String)$$

$$Dom(id, Thing)$$

$$Fun(id)$$

$$DiscCla(Person, Thing)$$

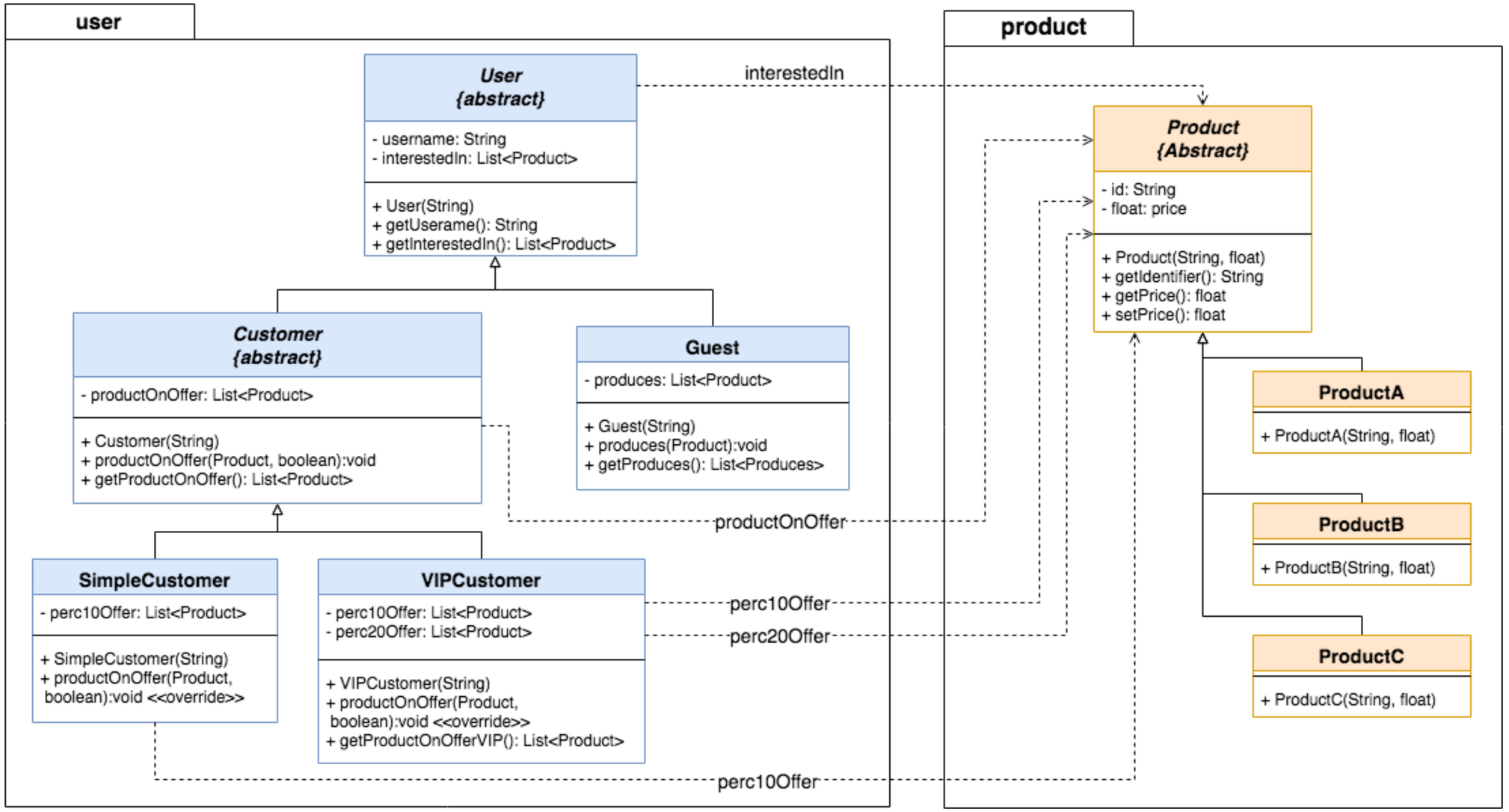
$$DiscCla(paint, sculpt, product)$$

DEMO - ART MARKET

- Idea of the app: catalog of artworks
- Load ontology
- Reasoning Routine on parsed axioms
- Attaching the application
- Application execution add new individuals but also makes assertion on individuals already present in the KB (consistency is maintained)
- **Not more active instances are discarded from ABox through OntologyHandler buffer**
- New axioms can be loaded
- DL queries

```
paints some Paint and sculpts some Sculpt
inverse crafts some (name value "Bernini"^^xsd:string)
```

DEMO - ESHOP


user
product
User
{abstract}
- username: String
- interestedIn: List<Product>
+ User(String)
+ getUserame(): String
+ getInterestedIn(): List<Product>
interestedIn
Customer
{abstract}
- productOnOffer: List<Product>
+ Customer(String)
+ productOnOffer(Product, boolean):void
+ getProductOnOffer(): List<Product>
Guest
- produces: List<Product>
+ Guest(String)
+ produces(Product):void
+ getProduces(): List<Produces>
productOnOffer
SimpleCustomer
- perc10Offer: List<Product>
+ SimpleCustomer(String)
+ productOnOffer(Product,
boolean):void <<override>>
VIPCustomer
- perc10Offer: List<Product>
- perc20Offer: List<Product>
+ VIPCustomer(String)
+ productOnOffer(Product,
boolean):void <<override>>
+ getProductOnOfferVIP(): List<Product>
perc10Offer
perc20Offer
perc10Offer
Product
{Abstract}
- id: String
- float: price
+ Product(String, float)
+ getIdentifier(): String
+ getPrice(): float
+ setPrice(): float
ProductA
+ ProductA(String, float)
ProductB
+ ProductB(String, float)
ProductC
+ ProductC(String, float)

DEMO - ESHOP

$$Guest \sqsubseteq User$$
$$Customer \sqsubseteq User$$
$$SimpleCurstomer \sqsubseteq Customer$$
$$VIPCustomer \sqsubseteq Customer$$
$$ProductA \sqsubseteq Product$$
$$ProductB \sqsubseteq Product$$
$$ProductC \sqsubseteq Product$$
$$Dom(interestedIn, User)$$
$$Rng(interestedIn, Product)$$
$$Dom(productOnOffer, Customer)$$
$$Rng(productOnOffer, Product)$$

$$Rng(username, String)$$
$$Dom(username, User)$$
$$Fun(name)$$
$$Rng(id, String)$$
$$Dom(id, Product)$$

$$Fun(id)$$
$$Rng(price, float)$$
$$Dom(price, Product)$$
$$Fun(price)$$

$$perc10Offer \dot{\sqsubseteq} productOnOffer$$
$$perc20Offer \dot{\sqsubseteq} productOnOffer$$
$$Dom(perc20Offer, VIPCustomer)$$
$$DiscCla(User, Product)$$
$$DiscCla(ProductA, ProductB, ProductC)$$
$$DisUni(User, Guest, Customer)$$
$$DiscCla(SimpleCustomer, VipCustomer)$$

$$PopularProductSC \sqsubseteq PopularProduct$$
$$PopularProductVC \sqsubseteq PopularProduct$$
$$PopularProductSC \equiv \geq 2\, interestedIn^{-}.SimpleCustomer$$
$$PopularProductVC \equiv \geq 2\, interestedIn^{-}.VIPCustomer$$
$$PopularProduct \equiv \geq 2\, interestedIn^{-}.User$$

DEMO - ESHOP

- Idea of the app is: **personal offers to customers** (if *vip* flag greater discount for vip customers)
- Random initialization of instances in the java application
- Ontologies provide **high-level concepts** not represented in the java application (e.g. queries in the reasoning routine)

```
PopularProduct
inverse interestedIn value username1 and inverse
perc10Offer value username1
inverse interestedIn value vip_username1 and inverse
perc10Offer value vip_username1
```

- **Jena ARQ Demo** (using the *owl/appOntologyES.owl* dump of the knowledge base)