* Java pattern matching statements case classes. Resource Monad hierarchy wrapping Case classes.
* Inputs / Sync Adapters:
* (Class, Instance, Member, Value) Events / Messages.
* Data Modelling Resources / Patterns (to / from Adapter Events):
* (Metaclass, Class, Instance, Context, Role, Occurrence);
* Models Patterns:
* (n-ary) Relations / Relationships. Foreign Keys. Dimensional / Measures, Master / Detail, Item / ItemDescription, Contexts / Roles, Interactions / Actors, Model / Facets (Actors / Roles Bindings. Property Graph, Others.
* Switch actions: Populate Models (RDFS, OWL, Sets, FCA Contexts, Functional MVC / DCI DOM / Others: SaILs). URNResource / Contexts / Roles.
* (Dimension, Unit, Measure, Value);
* Equivalence: Same Distance Measures.
* Entailment: Dimension, Unit, Measure Values entails other Measures / Values (Time, Speed, Distance). Contexts / Mappings.
* Composition of Case classes instances Augmentation via pattern matching (Aggregation: Kinds, Alignment: Contexts, Activation: Interactions). Map Reduce.
* Order: Kinds Hierarchies. Contexts Bindings Availability (Interactions State).
* Layers Case Classes Patterns / Resource Monads hierarchy:
* URN : (Class, ID);
* Class : Dimension, Unit (Attribute);
* ID : Measure, Value (Value);
* URN : ((Dimension, Unit), (Measure, Value));
* Resource : URN;
* URNResource : FunctionalResource (wrapper Monad).
* Transforms over URNResource wrapped URNs:
* C, S, P, O Resource Case Classes:
* Context : (C, \_, \_, \_) : Resource;
* Subject : (\_, S, \_, \_) : Resource;
* Predicate : (\_, \_, P, \_) : Resource;
* Object : (\_, \_, \_, O) : Resource;
* C, S, P, O : Resources;
* CSPOResource : FunctionalResource (wrapper Monad).
* Transforms over CSPOResource wrapped Resources:
* Statement Case Classes:
* Statement : (Resource, Resource, Resource, Resource) : Resource;
* CSPOStatement : (Context, Subject, Predicate, Object) : Statement;
* StatementResource : FunctionalResource (wrapper Monad).
* Transforms over StatementResource wrapped Statements:
* Subject(Predicate) : Object(s);
* Subject(Object) : Predicate(s);
* Predicate(Subject) : Object(s);
* Predicate(Object) : Subject(s);
* Object(Subject) : Predicate(s);
* Object(Predicate) : Subject(s);
* getContexts
* getSubjects
* getPredicates
* getObjects
* SPO Kinds Resource Case Classes:
* Kind : (URN, Resource, Attribute, Value) : Statement;
* SubjectKind : (URN, SubjectReifiedKind, Predicate, Object) : Statement;
* PredicateKind : (URN, PredicateReifiedKind, Subject, Object) : Statement;
* ObjectKind : (URN, ObjectReifiedKind, Predicate, Subject) : Statement;
* KindResource : Functional Resource (wrapper Monad).
* Transforms over (higher kinds reified) KindResource wrapped Statements:
* SubjectKind(PredicateKind) : ObjectKind(s);
* SubjectKind(ObjectKind) : PredicateKind(s);
* PredicateKind(SubjectKind) : ObjectKind(s);
* PredicateKind(ObjectKind) : SubjectKind(s);
* ObjectKind(SubjectKind) : PredicateKind(s);
* ObjectKind(PredicateKind) : SubjectKind(s);
* getSubjectKinds
* getPredicateKinds
* getObjectKinds
* SPO Occurrences Case Classes:
* Occurrence : (URN, Statement, Attribute, Value) : Statement;
* SubjectOccurrence : (URN, Statement, Predicate, Object) : Occurrence;
* PredicateOccurrence : (URN, Statement, Subject, Object) : Occurrence;
* ObjectOccurrence : (URN, Statement, Predicate, Subject) : Occurrence;
* OccurrenceResource : FunctionalResource (wrapper Monad).
* Transforms over OccurrenceResource wrapped Occurrences:
* Contexts Case Classes:
* Context : (URN, Statement, AttributeKind, ValueKind) : Statement;
* SubjectContext : (URN, Statement, PredicateKind, ObjectKind) : Context;
* PredicateContext : (URN, Statement, SubjectKind, ObjectKind) : Context;
* ObjectContext : (URN, Statement, PredicateKind, ObjectKind) : Context;
* ContextResource : FunctionalResource (wrapper Monad).
* Transforms over ContextResource wrapped Contexts:
* Mappings Case Classes:
* Mapping : (URN, Occurrence, AttributeKind, Value) : Statement;
* MappingResource : FunctionalResource (wrapper Monad).
* Transforms over MappingResource wrapped Mappings:
* TODO: Functional input data modelling alignments / patterns. Transforms. Dimensional alignment: Entailments. Relationships. Comparisons. Order.
* TODO:
* Order.
* Augmentations:
* Aggregation : Kinds / Contexts. Classification.
* Alignment : Occurrences / Kinds. Clustering (Attributes / Values inference)
* Activation : Occurrences / Mappings. (Value in Context. Prediction: Speed, Time : Distance)
* SemanticWebAlignmentTheory: Purpose driven data (assets), contexts (roles), interactions (actors) via semantic tags / labels /facets formal contexts.
* Qi4j / Elmo / Sesame RDF4J Alibaba: RDF Object Models.
* Model (Data):
* Categories / Assets
* Item / Inventory
* Master / Detail
* Facets / Properties
* Persistence: Index / SolrSail (Schema)
* Model Controllers (Contexts):
* Purpose: Purchase
* Roles: Buyerable, Purchaseable (Items)
* Model Properties: Hierarchies, Master / Detail, Facets, Schema, etc.
* Items Order: Intermediate Items (Contexts), Result.
* Model Views (Interactions):
* Session: aPurchase
* Actors: aBuyerable, aPurchaseable (Inventory)
* Model Properties: Hierarchies, Master / Detail, Facets, Schema, etc.
* Roles Order: Intermediate Roles (Interactions), Result.
* FCA Layers:
* Data: Attributes / Values x Types / Instances scaling. Price / Amount Attributes, Product / Item Objects.
* (Object, Attribute);
* Aggregated Measures: (Measure, Object, Attribute);
* (aMeasure, anObject, anAttribute);
* Information: Data across Dimensions Attributes / Values scaling. Time / Date / Store Price / Availability Attributes (Time / Date / Store Product Price Variation). Dimension aggregates axis Attributes / Values.
* (Object, Measures);
* Aggregated Dimensions: (Dimension, Objects, Measures);
* (Time / Date, anObject, (aMeasure: anObject, aDateAttribute));
* Knowledge: Dimensions Information Data Relationships Measures. (Product / Item, Date, Price Availability / Variation rate: Time / Store Product Price Percentage Variation Tendency). Example: Relationships aggregates Objects with same price variation tendency
* (Relationship, Dimension);
* Aggregated Relationships: (Relationship, Dimension, Measure);
* (PriceIncreaseRelationship, (aTimeDimension / Date / Interval, (aMeasure: aProduct, aProductPrice), priceIncreaseAttribute), aMeasure: anObject, anAttribute);
* Order Relationships:
* (PriceOrderingRelationship, (aPriceDimension / Price, (aMeasure: aProduct, aProductPrice), priceOrderAttribute), aMeasure: anObject, anAttribute);
* Cube Statement:
* (Relationship, Dimension, Measure);
* Relationships, Dimensions, Measures hierarchical Cube Traversal.
* Layers
* Data: Attributes / Values x Types / Instances scaling. Price / Amount Attributes, Product / Item Objects.
* Measures Mappings:
* (Object, Attribute);
* Aggregated Measures: (Measure, Object, Attribute);
* (aMeasure, anObject, anAttribute);
* Information: Data across Dimensions. Attributes / Values scaling. Time / Date / Store / Price / Availability Attributes. Attributes Variation. Dimension aggregates axis Objects / Attributes.
* Dimensional Mappings:
* (Measure, Attributes);
* Aggregated Dimensions: (Dimension, Measures, Attributes);
* (aTimeDimension  / Date / Interval, (aMeasure: anObject, anAttribute), anAttribute);
* Knowledge: Aggregates Dimensions Information Data Relationships. Example: Product Price Ordering / Percentage Variation Tendency across Date /  Time / Interval Dimension Measures.
* Relationship Mappings:
* (Dimension, Measures);
* Aggregated Relationships: (Relationship, Dimension, Measures);
* (PriceIncreaseRelationship, (aTimeDimension / Date / Interval, (aMeasure: aProduct, aProductPrice), priceIncreaseAttribute));
* Order Relationships:
* (PriceOrderingRelationship, (aPriceDimension / Price, (aMeasure: aProduct, aProductPrice), priceOrderAttribute));
* Mappings Matching: Relationships stated for matching Dimensions stated for matching Measures. Infer Data, Information, Knowledge  Relationship annotation / result Attributes.
* Cube Statement:
* (Relationship, Dimension, Measure);
* (Dimension, Unit, Measure, Value);
* Relationships, Dimensions, Measures hierarchical Cube Traversal.
* Encodings:
  + Scaling / Events Sourcing. Streams. (parse / populate SAIL to / from models: Sets, etc.).
  + Primes: Attributes Primes Sequence Product.
  + Bitstring: Attributes Bitstring Position Flags.
  + Base N: FCA Scaling by Layers of N Base aggregated Attributes (2, 4, 8, 16, 32, etc. Layers Base). Sub / super Attribute relationship. Dimensions / axis: Radices of multiple factors.
  + Objects / Attributes scaling: Layers Matching (merge contexts).
  + Data Aggregation Context: Metaclass / Class, CSPO / URNResource.
  + Information Alignment Context: Instance / Role, URNResource / Kind.
  + Knowledge Activation Context: Occurrence / Context, Kind / Statement.
  + Ordered Attributes:
  + Data Clustering: Measures (containment / distance).
  + Information Classification: generalizations / specializations (super / sub).
  + Knowledge Relationships (single / married, Friday / Saturday).
  + Order Encoding / Facets. ToDo.
  + Activation Function: Smaller Base / Larger Base Digits (sorted attributes) Aggregation.
* Models
  + Reference Model: FCA / Hashing. DIDs: URNs Resources. Events Sourcing.
  + URNs: URNFactor (ContextFactor, SubjectFactor, PredicateFactor, ObjectFactor);
  + Factors: URNs as CSPO Roles occurrence values product URNFactor (primes / bitstring). FCA Contexts Objects (Statements) / Attributes (URNs Roles Factors: bitstring / primes values hash lattice).
  + Augmentations: Matching / Inferencing. Upper / Matching URNs Role values Factors of Statements / URNs Role Factors.
  + Statement Aggregation Inference: Statement URN Roles Factors of URNs CSPO Factors values. Inferences embedded in Resources URNs Factors product of Statement URN Roles values.
* FCA Contexts:
* Statement (objects) CSPO Attributes:
* (URNFactor (ContextFactor x SubjectKindFactors : class, SubjectFactors : instance, PredicateFactors : attribute, ObjectFactors : value))
* Reified Statements (attributes), SPO Factors x Kinds (objects):
* (StatementFactors : SPO Factors x Provenance (SubjectFactors x SubjectKind, PredicateFactors x PredicateKind, ObjectFactors x ObjectKind))
* Provenance: Entailment. Provenance x reified SPO StatementFactors / source URNs. Statement entails / entailed by Factors product relations: transitive, reflexive, symmetrical (cause / effect, etc.).
* Alignment / Matching: Factors of matching URNs / Statements aggregated by product.
* Graph Statements (materialize / align) URNFactors attributes. Objects: CSPO Factor:
* (URNFactor, URNFactor, URNFactor, URNFactor)
* SubjectKind attributes. Objects: aggregated Subject Factors:
* (Context : super SKs factors, SubjectKind : matching factors / same Predicates Subject factors, Predicate, Object)
* PredicateKind attributes. Objects: aggregated Predicate Factors.
* (Context : super PKs factors, Subject, PredicateKind : matching factors / same Object Predicate factors, Object)
* ObjectKind attributes. Objects: aggregated Object Factors.
* (Context : super OKs factors, Subject, Predicate, Object : matching factors / same Subject Object factors)
* Order / Flow Mappings attributes. Objects: aggregated State Factors.
* (Context, SubjectKind, PredicateKind, ObjectKind)
* Context: State factor (Kinds) previous / next Kinds Mapping Layout. SPO Kinds State (Kinds factors). Super / Sub Context Alignment. Kinds "joins": materialize / align order / flow Statements. Kinds flow: Statement Kinds / SPO States (Contexts). Order: upper / super / sub hierarchies (Kinds Contexts / factors). Joins matching Kinds factors (flows). Provenance state flows.
  + Sets. Data Aggregation.
  + ResourceURNs (Occurrences). Contexts Activation.
  + DOM (Dynamic Object Model Object Graph Mapper): Functional Dataflow. OpenRDF Sesame Elmo / Alibaba. Interactions Alignments.
* Augmentations
  + Aggregation.
  + Activation.
  + Alignment.
* Ontology Matching: FCA / Hashing
  + Data. CSPO.
  + Schema. Kinds.
  + Behavior. Contexts.
* Inferences: FCA / Hashing
  + Data Ontology Matching.
  + Contexts Ontology Matching.
  + Interactions Ontology Matching.
* DDD: Forms Dialogs Protocol: Suggestions (infer Purposes from Gestures: DCI Matching)
* Detectar mediante lo que yo llamo "Gestures" el Propósito de una Interacción de un usuario (persona o servicio: flujos de navegación) en un Contexto dado, según los Datos del "diálogo" en un protocolo que permita inferir y facilitar la intención o el objetivo de la Interacción mediante "Suggestions".
* Detect by means of user "Gestures" (person or service browsing flows) the Purpose of an Interaction, in a given Context, following Data of a of a "dialog" in a protocol such that the Interaction intention or objectives may be inferred and guided by means of "Suggestions".