ROBBIE'S WORLD – the FACTS: Robbie needs to KNOW **E V E R Y T H I N G** about his world – either as memorized facts, or, preferably, as something he can reason out from his rules in his KB. This includes:

1) what things EXIST in the world: the table, Robbie's hand, all the different objects
   - things need names so they can be referred to, like: theBall, theGlass, blockC, ..., robbie
   But do NOT specifically define what exists in the fact DB, since Robbie can infer that something exists using his knowledge (i.e., a rule in his KB) - he KNOWS something exists IF it . . .).
   The name of the thing will be a parameter in both the static DB and the dynamic DB (to be able to link the two sets of properties).

2) (static DB) PROPERTIES of each INDIVIDUAL thing (e.g., theBall, theGlass) that can NOT change (and that are NOT specific to this particular physical arrangement of the things).
   For example, the TYPE of thing the object is (generally referred to with the "isa" predicate), its SIZE, its COLOR, . . ., etc.
   In Prolog one might use: `isa(theDish,aDish).` But because we're using record-based facts with translator rules, you should instead include the FACT:
   `objectStaticProperties(theDish,aDish,large,white,...).`
   with the a translator rule such as:
   ```
   isa(ObjectName,Type) :-
       objectStaticProperties(ObjectName,Type,_,_,...).
   ```
   Do NOT include properties of TYPES of things e.g., aBall, aGlass in the DB.
   That information should be the KB not in the DB – for example, include:
   "The dish is a type of a dish" [="THE dish is A dish" ="THIS dish is a type of dish"]
   But don't include "A dish is a type of eating utensil" – that'd be knowledge, not data.
   Similarly, the following is knowledge and so goes in the KB:
   "A dish is made of china" [= "ALL dishes are made of china" = "Dishes are china"]
   in Prolog:
   ```
   madeOf(Thing,china) :-
       isa(Thing,aDish).
   ```
   [NOTE that the rule does NOT mention any specific object by name (i.e., theDish) – it only describes types of objects (i.e., aDish).
   Robbie will be able to use reasoning to determine that THE dish is made of china" because he knows this fact & rule:
   "THE dish is a type of A dish" and "A dish is made of china"
   Therefore "THE dish is made of china".
   Similarly: The marble is a marble. (a FACT, after translation)
   All marbles are spheres. (a KB RULE) (If it's a marble, then it's a sphere).
   All spheres have round edges. (a KB RULE) (If it's a sphere, then it has round edges).
   THEREFORE (Robbie reasons/deduces/infers) that theMarble has round edges, without having memorized that as a FACT.
   Consider: Can this particular marble (theMarble) roll? Robbie can infer this:
   Something can roll IF it has a round edge AND
   (for cylinders or cones) it's lying on its side
   (but for spheres) it doesn't matter what its orientation is
   . . .
   [NOTE: CAN roll (now, as it is) is different that COULD roll (potentially) in the KB]

   NOTE about MATERIAL as a property.
   Some TYPES of things are always made of a particular material (for our world) as a rule:
   All prisms, marbles and glasses are made of glass. Dishes are made of china.
   All blocks and pencils are wood. Robbie is metal.
   Balloons are always made of rubber.
   Other TYPES of things vary – so specify a material for that particular thing as a fact for that thing:
   The table and the cube are metal. The cup is made of paper.
   The baseball is made of leather (let's say). The ball is rubber.
   The box and tube are cardboard. The rock is granite.

3) (dynamic DB) PROPERTIES of each INDIVIDUAL thing - specific to this particular world
   - the orientation of each object now – e.g., on its base or on its side or on its top
   - KB rules, not facts, should handle the "symmetry" situation for various shapes
   - you'll have to refer to things' names as a parameter to be able to tie them to static facts later
   - objects and Robbie need dynamic properties, but the table doesn't since it can't change

4) (dynamic DB) RELATIONSHIPS between things (for this specific configuration of the world)
   For objects:
   - which object is left of (meaning directlyLeftOf) which other object
     (not somewhereLeft which can be inferred using the KB)
     NOTE: only include left data for things actually directly on the table
     (since data about leftness for thePrism, etc. can be inferred from the KB)
   - which object is on (meaning directlyOn) which other thing (an object or the table)
     NOTE: on means actually sitting on top of, touching the lower thing
   - CAUTION – keep these facts as minimal as possible
     Put rules in the KB for reasoning about somewhereLeft, somewhereAbove, . . . directlyRight, somewhereRight, . . .
   For Robbie:
   - you don't need left/on parameters, but store what he's holding (if anything)
   For the table:
   - you don't need any dynamic DB facts
   NOTE: for the dynamic DB, combine categories #3 & #4 parameters in single record-based facts
   `objectDynamicProperties(theDish, onBase, theRock, table).`
   NOTE: Robbie's facts and the objects' facts will have different number of parameters