Below is a summary of each class and its key methods from the provided code:

**AvailPartWrapper2**

* **Purpose:**  
  Wraps around a game part and provides additional information needed by the application.
* **Properties:**
  + **NameID:** Public property holding the part’s identifier (derived from the part’s name).
  + **Experiments:** Public list of experiment IDs assigned to the part.
* **Internal Fields:**
  + **part:** An instance of AvailablePart representing the actual part.
  + **partTitle:** The title of the part, determined via a helper method.
  + **numAvailable:** An integer tracking the number available (initially 0).
  + **scanType:** An enum value (SCANsatSCANtype) indicating the scan type if applicable.
  + **scanSatPart:** A boolean flag set to true if the part is associated with a scan type.
* **Constructor:**  
  Initializes the wrapper by setting NameID, storing the provided part and scan type, and computing the part title via the static helper.
* **Static Method:**
  + **GetPartTitle(string name):**  
    Retrieves the part’s title based on its name (replacing underscores, checking purchase status, etc.).

**Experiment**

* **Purpose:**  
  Wraps around an experiment and enriches it with additional metadata.
* **Properties:**
  + **ExperimentID:** Public property holding the experiment’s identifier.
  + **Name:** Public property for the experiment’s name.
* **Internal Fields:**
  + **scanType:** Holds a SCANsatSCANtype value if the experiment is related to scan data.
  + **scanSatExperiment:** Boolean flag indicating if the experiment is a SCANsat experiment.
  + **experimentTitle:** The title of the experiment, determined via a helper method.
  + **contractGuid:** The Guid that links the experiment to a specific contract.
* **Methods:**
  + **ContractExperiment():**  
    Returns a string that combines the contract’s Guid and the experiment ID.
* **Constructor:**  
  Accepts a CEP\_Key\_Tuple and an optional scansatExpID string. Depending on the presence of scansatExpID, it sets up the experiment’s scan-related fields and computes the experiment title using a private helper method.
* **Private Method:**
  + **getExpTitle(string name):**  
    Retrieves the title for an experiment by querying the experiment data (using the ResearchAndDevelopment API).

**CheckModule**

* **Purpose:**  
  Represents a check module with descriptive information.
* **Properties:**
  + **Types:** A string indicating the type of check module.
  + **Description:** A string providing a description of the check module.

**Param**

* **Purpose:**  
  Encapsulates parameter data used in contracts, including various lists for grouping related data.
* **Properties:**
  + **Key & Value:** Basic key/value pairs.
  + **KerbalName:** A string specifying the Kerbal (character) associated with the parameter.
* **Lists (Collections):**
  + **RequestedParts:** List of part group keys that are requested.
  + **Vessels:** List of vessel IDs.
  + **CheckModules:** List of CheckModule objects (each with its type and description).
  + **PartNames:** List of part names.
  + **ModuleNames:** List of module names.

**ContractWrapper**

* **Purpose:**  
  Wraps around a contract and holds all related data such as experiments, parts, and parameters.
* **Properties:**
  + **Id:** The contract’s unique Guid.
  + **SortOrder:** A string used to determine the order in which contracts are displayed.
* **Internal Fields:**
  + **selected & active:** Boolean flags for UI or state tracking.
  + **contractContainer:** An internal container object holding more detailed contract data.
* **Collections:**
  + **ExperimentGroups:** A dictionary mapping group keys to lists of experiments.
  + **PartGroupKeys:** A list of part group keys assigned to the contract.
  + **Params:** A list of Param objects containing various parameters for the contract.
* **Constructor:**  
  An internal constructor that initializes the contract from a contractContainer instance, setting the Id and default state flags.

**Repository**

* **Purpose:**  
  Acts as a centralized manager for contracts, part groups, experiments, and parameters, providing methods to add, assign, and retrieve data.
* **Collections:**
  + **PartGroups:** A dictionary where each key (string) maps to a list of AvailPartWrapper2 instances.
  + **Contracts:** A dictionary where each key (Guid) maps to a ContractWrapper.
* **Key Methods:**
  + **AddContract(ContractWrapper contract):**  
    Adds a new contract to the repository, ensuring no duplicate Id exists.
  + **AddExperimentGroupToContract(Guid contractId, string groupKey, IEnumerable experiments):**  
    Associates a new experiment group with a given contract.
  + **AddExperimentToContract(Guid contractId, string groupKey, Experiment experiment):**  
    Adds a single experiment to an existing group or creates a new group within a contract.
  + **AddParts(string groupKey, IEnumerable parts):**  
    Adds a new group of parts to the repository.
  + **AddPart(int from, string groupKey, AvailPartWrapper2 part):**  
    (Provided in the extended version) Adds a single part to a group, logging an error if the part’s NameID is null.
  + **AssignExperimentToPart(string partGroupKey, string partNameID, string experimentID):**  
    Assigns a single experiment to a part within a specified part group.
  + **AssignExperimentsToPart(string partGroupKey, string partNameID, IEnumerable experimentIDs):**  
    Assigns multiple experiments to a part.
  + **AddPartGroupToContract(Guid contractId, string partGroupKey):**  
    Associates an existing part group from the repository with a given contract.
  + **GetPartsForContractFromPartGroups(Guid contractId):**  
    Retrieves all parts from the part groups that have been added to the contract.
  + **GetPartsForContract(Guid contractId):**  
    Retrieves parts for a contract based on the experiments associated with that contract.
  + **GetPartsForExperimentInContract(Guid contractId, string experimentID):**  
    Retrieves parts that are linked to a specific experiment within a contract.
  + **GetExperimentsInContract(Guid contractId):**  
    Returns a flat list of all experiments in the specified contract.
  + **GetContracts(bool sorted = false):**  
    Returns a list of contracts; if the sorted parameter is true, contracts are sorted by SortOrder and then by Id.