```
class DesignatorDescription(ABC):
    """
    :ivar resolve: The specialized_designators function to use for this designator, defaults to self.ground
    """

    def __init__(self, resolver: Optional[Callable] = None):
        """
        Create a Designator description.
        """
        if resolver is None:
            self.resolve = self.ground

    def ground(self) -> Any:
        """
        Should be overwritten with an actual grounding function which infers missing properties.
        """

class ObjectDesignatorDescription(DesignatorDescription):
    """
    Class for object designator descriptions.
    Descriptions hold possible parameter ranges for object designators.
    """

    @dataclass
    class Object:
        """
        A single element that fits the description.
        """

        name: str
        """
        Name of the object
        """

        obj_type: ObjectType
        """
        Type of the object
        """

        world_object: Optional[WorldObject]
        """
        Reference to the World object
        """

        _pose: Optional[Callable] = field(init=False)
        """
        A callable returning the pose of this object. The _pose member is used overwritten for data copies
        which will not update when the original world_object is moved.
        """
    def __init__(self, names: Optional[List[str]] = None, types: Optional[List[ObjectType]] = None,
                 resolver: Optional[Callable] = None):
        """
        Base of all object designator descriptions. Every object designator has the name and type of the
object.

        :param names: A list of object names
        :param types: A list of object types
        """

    def ground(self) -> Union[Object, bool]:
        """
        Return the first object from the world that fits the description.
```