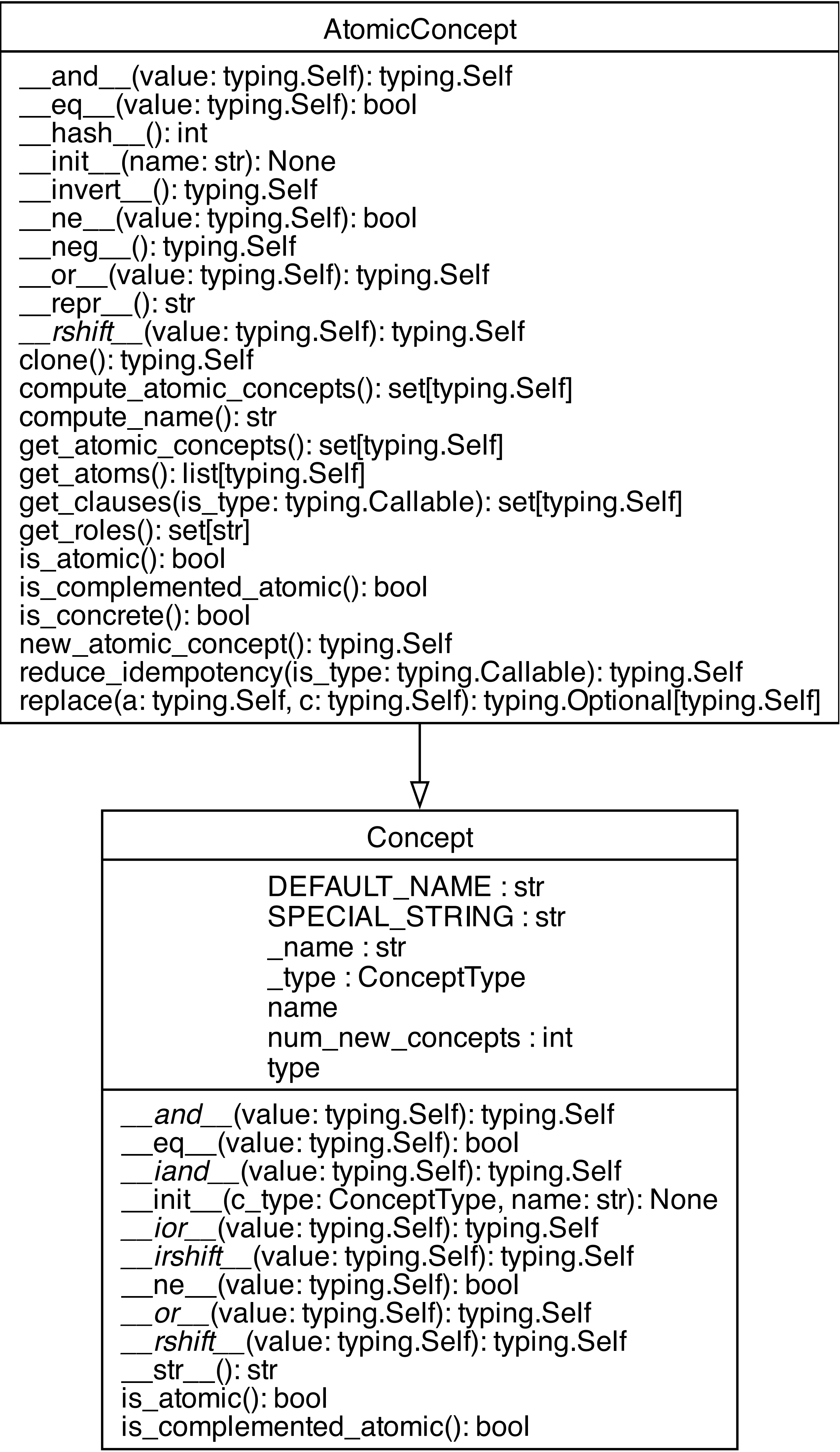

```
AtomicConcept
__and__(value: typing.Self): typing.Self
__eq__(value: typing.Self): bool
__hash__(): int
__init__(name: str): None
__invert__(): typing.Self
__ne__(value: typing.Self): bool
__neg__(): typing.Self
__or__(value: typing.Self): typing.Self
__repr__(): str
__rshift__(value: typing.Self): typing.Self
clone(): typing.Self
compute_atomic_concepts(): set[typing.Self]
compute_name(): str
get_atomic_concepts(): set[typing.Self]
get_atoms(): list[typing.Self]
get_clauses(is_type: typing.Callable): set[typing.Self]
get_roles(): set[str]
is_atomic(): bool
is_complemented_atomic(): bool
is_concrete(): bool
new_atomic_concept(): typing.Self
reduce_idempotency(is_type: typing.Callable): typing.Self
replace(a: typing.Self, c: typing.Self): typing.Optional[typing.Self]
```

```
Concept
DEFAULT_NAME : str
SPECIAL_STRING : str
_name : str
_type : ConceptType
name
num_new_concepts : int
type

__and__(value: typing.Self): typing.Self
__eq__(value: typing.Self): bool
__iand__(value: typing.Self): typing.Self
__init__(c_type: ConceptType, name: str): None
__ior__(value: typing.Self): typing.Self
__irshift__(value: typing.Self): typing.Self
__ne__(value: typing.Self): bool
__or__(value: typing.Self): typing.Self
__rshift__(value: typing.Self): typing.Self
__str__(): str
is_atomic(): bool
is_complemented_atomic(): bool
```
]