```
import json
import re
from typing import Type, TypeVar

import yaml
from pydantic import BaseModel

T = TypeVar("T", bound=BaseModel)


class YamlParsingException(Exception):
    """Custom exception for errors in YAML parsing."""


class YamlOutputParser:
    """Parse YAML output using a Pydantic model.

    This parser is designed to extract YAML formatted data from a given string
    and parse it using a specified Pydantic model for validation.

    Attributes:
        pydantic_object: A Pydantic model class for parsing and validation.
        pattern: A regex pattern to match YAML code blocks.


    Examples:
```

```
>>> from pydantic import BaseModel
>>> from swarms.utils.yaml_output_parser import YamlOutputParser
>>> class MyModel(BaseModel):
...     name: str
...     age: int
...
>>> parser = YamlOutputParser(MyModel)
>>> text = "```yaml\nname: John\nage: 42\n```"
>>> model = parser.parse(text)
>>> model.name

"""

def __init__(self, pydantic_object: Type[T]):
    self.pydantic_object = pydantic_object
    self.pattern = re.compile(
        r"^```(?:ya?ml)?(?P<yaml>[^`]*)", re.MULTILINE | re.DOTALL
    )

def parse(self, text: str) -> T:
    """Parse the provided text to extract and validate YAML data.

    Args:
        text: A string containing potential YAML data.

    Returns:
```

```
        An instance of the specified Pydantic model with parsed data.

    Raises:
        YamlParsingException: If parsing or validation fails.
    """
    try:
        match = re.search(self.pattern, text.strip())
        yaml_str = match.group("yaml") if match else text

        json_object = yaml.safe_load(yaml_str)
        return self.pydantic_object.parse_obj(json_object)

    except (yaml.YAMLError, Exception) as e:
        name = self.pydantic_object.__name__
        msg = (
            f"Failed to parse {name} from text '{text}'."
            f" Error: {e}"
        )
        raise YamlParsingException(msg) from e

def get_format_instructions(self) -> str:
    """Generate formatting instructions based on the Pydantic model schema.

    Returns:
        A string containing formatting instructions.
    """
```

```
schema = self.pydantic_object.schema()
reduced_schema = {
    k: v
    for k, v in schema.items()
    if k not in ["title", "type"]
}
schema_str = json.dumps(reduced_schema, indent=4)

format_instructions = (
    f"YAML Formatting Instructions:\n{schema_str}"
)
return format_instructions
```