```
from typing import List
from .error_types import CompilerError

class ErrorReporter:
    @staticmethod
    def print_errors(errores: List[CompilerError], warnings: List[CompilerError],
show_warnings: bool = True):
        """Imprime todos los errores y advertencias de forma legible"""
        if errores:
            print("\n=== ERRORES ===")
            for error in errores:
                print(f"❌ {error}")

        if warnings and show_warnings:
            print("\n=== ADVERTENCIAS ===")
            for warning in warnings:
                print(f"⚠  {warning}")

        print(f"\nResumen: {len(errores)} error(es), {len(warnings)}
advertencia(s)")

    @staticmethod
    def generate_error_report(errores: List[CompilerError], warnings:
List[CompilerError]) -> str:
        """Genera un reporte completo de errores en formato de texto"""
        report = []

        if errores:
            report.append("=== ERRORES ===")
            for error in errores:
                report.append(f"[ERROR] {error}")

        if warnings:
            report.append("\n=== ADVERTENCIAS ===")
            for warning in warnings:
                report.append(f"[WARNING] {warning}")

        report.append(f"\nTotal: {len(errores)} error(es), {len(warnings)}
advertencia(s)")

        return "\n".join(report)

    @staticmethod
    def format_error_with_context(error: CompilerError, source_code: str) -> str:
        """Formatea un error mostrando el contexto en el código fuente"""
        if error.token and hasattr(error.token, 'posicion'):
            # Encontrar la línea donde ocurrió el error
            lines = source_code.split('\n')
            if error.linea <= len(lines):
                context_line = lines[error.linea - 1]
```

```
        pointer = ' ' * (error.columna - 1) + '^'
        return f"{error}\n{context_line}\n{pointer}"

return str(error)
```

```
from .error_types import NivelError, FaseError, ErrorCode, CompilerError,
ERRORES_LEXICOS, ERRORES_SINTAXIS, ERRORES_SEMANTICOS
from .error_handler import ErrorHandler
from .error_reporter import ErrorReporter

__all__ = [
    'NivelError', 'FaseError', 'ErrorCode', 'CompilerError', 'ERRORES_LEXICOS',
'ERRORES_SINTAXIS', 'ERRORES_SEMANTICOS', 'ErrorHandler', 'ErrorReporter'
]
```