Universidade Federal do Ceará - Departamento de Computação
Manutenção e Evolução de Software (CK0221) | Manutenção e Evolução de Software (CKP9000)
Professor: Lincoln Souza Rocha | E-mail: lincon@dc.ufc.br | Semestre: 2024-1

Tarefa Prática sobre Engenharia Reversa de Código

O objetivo dessa tarefa prática é exercitar os métodos de engenharia reversa de código usando a ferramenta Spoon (http://spoon.gforge.inria.fr). Para isso, você deverá utilizar como sistema alvo o projeto BankSys (https://github.com/lincolnrocha/bank-sys).

Como instanciar a API do Spoon

```
SpoonAPI spoonAPI = new Launcher();
spoonAPI.addInputResource("project_src_path");
spoonAPI.buildModel();
```

Como instanciar a API do Spoon para usar processors

```
SpoonAPI spoonAPI = new Launcher();
spoonAPI.addInputResource("target_project_src_path");
spoonAPI.buildModel();
spoonAPI.addProcessor("br.ufc.LoCExtractor");
spoonAPI.process();
```

Como filtrar elementos do tipo CtClass à partir da API

```
for(CtClass c : spoonAPI.getElements(new TypeFilter<CtClass>(CtClass.class))) {
  (...)
}
```

Como implementar um processor para o tipo CtClass

```
public class LoCExtractor extends AbstractProcessor<CtClass<?>> {


 @Override
 public void process(CtClass<?> element) {
     String qualifiedName = element.getQualifiedName();
     int startLine = element.getPosition().getLine();
     int endLine = element.getPosition().getEndLine();
     double totalLoC = endLine - startLine;
    (...)
 }
}
```

Como obter todos os tipos (classes, interfaces, enum, etc) do projeto e suas as respectivas referências.

```
Collection<CtType<?>> types = spoonAPI.getModel().getAllTypes();


for(CtType<?> type : types) {
  if(!type.isShadow()) {
    for (CtTypeReference<?> referredType : type.getReferencedTypes()) {
      if(!referredType.isShadow()){
        (...)
      }
    }
}
```

1) Escreva um programa usando Spoon que extraia a lista de adjacência do projeto BankSys. Você deve usar a abordagem query do Spoon. Você deve armazenar as dependências em um HashMap<String,Set<String>> onde o primeiro parâmetro é o nome qualificado do tipo (classe, interface, enum, etc) em análise e segundo é o conjunto de nomes qualificados dos tipos que são adjacentes ao tipo em análise, i.e.; tipos do sistema BankSys referenciados pelo tipo em questão. Ao final, você deve prover uma impressão da lista de adjacência.

2) Escreva um programa usando Spoon que extraia a lista de adjacência do projeto BankSys considerando apenas as relações de dependência de herança entre os tipos. Você deve usar a abordagem query do Spoon. Você deve armazenar as dependências em um HashMap<String,Set<String>> onde o primeiro parâmetro é o nome qualificado do tipo (classe, interface, enum, etc) em análise e segundo é o conjunto de nomes qualificados dos tipos que são adjacentes ao tipo em análise, i.e.; tipos do sistema BankSys que possuem uma relação de herança com o tipo em questão. Ao final, você deve prover uma impressão da lista de adjacência.

3) Escreva um programa usando Spoon que extraia a lista de adjacência do projeto BankSys considerando apenas as relações de dependência de chamada de métodos entre os tipos. Você deve usar a abordagem query do Spoon. Você deve armazenar as dependências em um HashMap<String,Set<String>> onde o primeiro parâmetro é o nome qualificado do tipo (classe, interface, enum, etc) em análise e segundo é o conjunto dos métodos chamados seguindo o formato: nome qualificado do tipo chamado + "#" + assinatura do método chamado, i.e.; métodos dos tipos do sistema BankSys que são chamados pelo tipo em análise. Ao final, você deve prover uma impressão da lista de adjacência.

4) Escreva um programa usando Spoon que extraia um conjunto de métricas de cada tipo do projeto BankSys usando a abordagem de processors. As seguintes métricas devem ser extraída por cada processor:
   1. LoC (LoCExtractor): número de linhas de código do tipo em análise;

2. Fan-In (FanInExtractor): número de tipos que referenciam o tipo em análise;
3. Fan-Out (FanOutExtractor): número de tipos que o tipo em análise referência;
4. NoDM (NoDMExtractor): número de métodos declarados no tipo em análise;
5. NoHM (NoHMExtractor): número de métodos herdados pelo tipo em análise;
6. NoPrM (NoPrMExtractor): número de métodos privados do tipo em análise;
7. NoPuM (NoPuMExtractor): número de métodos públicos do tipo em análise;
8. NoDF (NoDFExtractor): número de atributos declarados no tipo em análise;
9. NoFH (NoHFExtractor): número de atributos herdados pelo tipo em análise;
10. NoPrF (NoPrFExtractor): número de atributos privados do tipo em análise;
11. NoPuF (NoPuFExtractor): número de atributos públicos do tipo em análise;
12. NoC (NoCExtractor): número de tipos que herdam do tipo em análise.