preliminary

assign_stmt

skeleton: target_variable "=" rvalue

- if rvalue = array_ref

```
str = "assign an array" + rvalue + "to a variable" + target_variable
```

- if rvalue = constant

```
str = "assign a constant value" + rvalue + "to a variable" + target_variable
```

- if rvalue = instance_field_ref

```
str = "assign an field instance" + rvalue + "to a variable" + target_variable
```

- if rvalue = local

```
str = "assign a local variable" + rvalue + "to a variable" + target_variable
```

- if rvalue = next_next_stmt_address

```
str = "assign the address of statement" + rvalue + "to a variable" + target_variable
```

- if rvalue = static_field_ref

```
str = "assign a statc field" + rvalue + "to a variable" + target_variable
```

- if rvalue = binop_expr

```
str = "assign the outcome of binary operatiion" + rvalue + "to a variable" + target_variable
```

- if rvalue = cast_expr

```
str = "the type of local variable" + rvalue + "is cast to" + className + "and the value of local
variable" + rvalue + "is assigned to a local variable" + target_variable + "."
```

- if rvalue = invoke_expr

> if invoke_expr = interface_invoke

```
str = "assign a value to local variable" + target variable + "."
  str += caller + "invokes an interface method from interface " + className + "."
  str += "the method has name as " + methodName + ","
  str += "and has parameter types as "
  for paraType in paraTypes:
      str += paraType + ","
  str += "and is fed with parameter values as "
  for paraValue in paraValues:
      str += paraValue + ","
  str += "and returns a value of type" + retType + "."
  str += "finally the returned value of type " + retType + "is assigned to " + target variable
+ "."
```

> if invoke_expr = static_invoke

```
str = "assign a value to local variable" + target variable + "."
  str += caller + "invokes an static method from class " + className + "."
  str += "the method has name as " + methodName + ","
  str += "and has parameter types as "
  for paraType in paraTypes:
      str += paraType + ","
  str += "and is fed with parameter values as "
  for paraValue in paraValues:
      str += paraValue + ","
  str += "and returns a value of type" + retType + "."
  str += "finally the returned value of type " + retType + "is assigned to " + target variable
+ "."
```

> if invoke_expr = virtual_invoke

```
str = "assign a value to local variable" + target variable + "."
  str += caller + "invokes an virtual(instance) method from class " + className + "."
  str += "the method has name as " + methodName + ","
  str += "and has parameter types as "
  for paraType in paraTypes:
      str += paraType + ","
  str += "and is fed with parameter values as "
  for paraValue in paraValues:
      str += paraValue + ","
  str += "and returns a value of type" + retType + "."
  str += "finally the returned value of type " + retType + "is assigned to " + target variable
+ "."
```

> if invoke_expr = special_invoke

```
    str = "assign a value to local variable" + target variable + "."
      str += caller + "invokes an special (init, private, inherited) method from class " +
    className + "."
      str += "the method has name as " + methodName + ","
      str += "and has parameter types as "
      for paraType in paraTypes:
          str += paraType + ","
      str += "and is fed with parameter values as "
      for paraValue in paraValues:
          str += paraValue + ","
      str += "and returns a value of type" + retType + "."
      str += "finally the returned value of type " + retType + "is assigned to " + target variable
    + "."
```

invoke_stmt

> if invoke_stmt = interface_invoke:

```
str = caller + " invokes an interface method from interface " + className + ". The method has name
as " + methodName + ", has parameter types as "
for param_type in param_types:
    str += param_type + ", "
str += "and are feed with parameter values as "
for param_feed in param_feeds:
    str += param_feed + ", "
str += "Finally, the method returns as type " + ret_type
```

> if invoke_stmt = static invoke:

```
str = caller + " invokes an static method from class " + className + ". The method has name as " +
methodName + ", has parameter types as "
for param_type in param_types:
    str += param_type + ", "
str += "and are feed with parameter values as "
for param_feed in param_feeds:
    str += param_feed + ", "
str += "Finally, the method returns as type " + ret_type
```

> if invoke_stmt = virtual_invoke:

```
str = caller + " invokes an virtual (instance) method from class " + className + ". The method has
name as " + methodName + ", has parameter types as "
for param_type in param_types:
    str += param_type + ", "
str += "and are feed with parameter values as "
for param_feed in param_feeds:
    str += param_feed + ", "
str += "Finally, the method returns as type " + ret_type
```

> if invoke_stmt = special_invoke:

```
str = caller + " invokes an special (init, private, inherited) method from class " + className +
". The method has name as " + methodName + ", has parameter types as "
for param_type in param_types:
    str += param_type + ", "
str += "and are feed with parameter values as "
for param_feed in param_feeds:
    str += param_feed + ", "
str += "Finally, the method returns as type " + ret_type
```

breakpoint_stmt

```
str = "break point"
```

entor_monitor_stmt

```
str = "enter monitor"
```

goto_stmt

```
str = "go to" + destination
```

if_stmt =

```
str = "if" + condition
```

lookup_swtich_stmt

```
str = "look up switch"
```

nop_stmt

```
str = "nop"
```

return_stmt

```
str = "return a value of type " + retType
```

return_void_stmt

```
str = "return a void value"
```

throw_stmt

```
str = "throws a value of type" + type
```