Formal Definition of SWIN

Chenglong Wang Jun Li Yingfei Xiong Zhenjiang Hu

1 Featherweight Java

1.1 Syntax

Class Declaration

$$\mathtt{CL} ::= \mathtt{class\ C\ extends\ C}\{\bar{\mathtt{C}}\ \bar{\mathtt{f}}; \mathtt{K}\ \bar{\mathtt{M}}\}$$

Constructor Declaration

$$\mathtt{K} ::= \mathtt{C}\ (\bar{\mathtt{C}}\ \bar{\mathtt{f}})\ \{\mathtt{super}(\bar{\mathtt{f}}); \mathtt{this}.\bar{\mathtt{f}} = \bar{\mathtt{f}}\}$$

Method Declaration

$$\mathtt{M} ::= \mathtt{C\ m}(\bar{\mathtt{C}}\ \bar{\mathtt{x}})\ \{\mathtt{return\ t};\}$$

Term

$$\mathtt{t} ::= \mathtt{x} \mid \mathtt{t.f} \mid \mathtt{t.m}(\bar{\mathtt{t}}) \mid \mathtt{new\ C}(\bar{\mathtt{t}}) \mid (\mathtt{C})\ \mathtt{t}$$

1.2 Type System

1.2.1 Subtyping

$$\frac{}{\mathtt{C} <: \mathtt{C}}\ \text{(S-SELF)} \qquad \frac{\mathtt{C} <: \mathtt{D} \quad \mathtt{D} <: \mathtt{E}}{\mathtt{C} <: \mathtt{E}}\ \text{(S-TRANS)}$$

$$\frac{CT(\mathtt{C}) = \mathtt{class\ C\ extends\ D}\ \{...\}}{\mathtt{C} <: \mathtt{D}}\ \text{(S-DEF)}$$

1.2.2 Auxiliary Functions

$$\frac{}{\mathtt{fields}(\mathtt{Object}) = \{\}}\ \text{FIELD-OBJECT}$$

$$\frac{\begin{array}{c} CT(\mathtt{C}) = \mathtt{class\ C\ extends\ D}\ \{\bar{\mathtt{C}}\ \bar{\mathtt{f}};\ \mathtt{K}\ \bar{\mathtt{M}}\} \\ \mathtt{fields}(\mathtt{D}) = \bar{\mathtt{D}}\ \bar{\mathtt{g}} \end{array}}{\mathtt{fields}(\mathtt{C}) = \bar{\mathtt{D}}\ \bar{\mathtt{g}},\ \bar{\mathtt{C}}\ \bar{\mathtt{f}}}\ \text{(FIELD-LOOKUP)}$$

$$\frac{\begin{array}{c} CT(\mathtt{C}) = \mathtt{class\ C\ extends\ D}\ \{\bar{\mathtt{C}}\ \bar{\mathtt{f}};\ \mathtt{K}\ \bar{\mathtt{M}}\} \\ \mathtt{B\ m}\ (\bar{\mathtt{B}}\ \bar{\mathtt{x}})\ \{\mathtt{return\ t};\} \in \bar{\mathtt{M}} \end{array}}{\mathtt{mtype}(\mathtt{m}, \mathtt{C}) = \bar{\mathtt{B}} \to \mathtt{B}}\ \text{(METHOD-LOOKUP1)}$$

$$\frac{\begin{array}{c} \mathtt{CT}(\mathtt{C}) = \mathtt{class\ C\ extends\ D}\ \{\bar{\mathtt{C}}\ \bar{\mathtt{f}};\ \mathtt{K}\ \bar{\mathtt{M}}\} \\ \mathtt{m}\ \text{is not defined in}\ \bar{\mathtt{M}} \end{array}}{\mathtt{mtype}(\mathtt{m}, \mathtt{C}) = \mathtt{mtype}(\mathtt{m}, \mathtt{D})}\ \text{(METHOD-LOOKUP2)}$$

$$\frac{\mathtt{mtype}(\mathtt{m}, \mathtt{D}) = \bar{\mathtt{D}} \to \mathtt{D}_0\ \mathtt{implies}\ \bar{\mathtt{C}} = \bar{\mathtt{D}}\ \mathtt{and}\ \mathtt{C}_0 = \mathtt{D}_0}{\mathtt{override}(\mathtt{m}, \mathtt{D}, \bar{\mathtt{C}} \to \mathtt{C}_0)}\ \text{(OVERRIDE)}$$

1.2.3 Typing

$$\frac{\mathtt{x} : \mathtt{C} \in \Gamma}{\Gamma \vdash \mathtt{x} : \mathtt{C}} \text{ (FJ-VAR)}$$

$$\frac{\Gamma \vdash \mathtt{t}_0 : \mathtt{C}_0 \qquad \mathtt{fields}(\mathtt{C}_0) = \bar{\mathtt{C}}\ \bar{\mathtt{f}}}{\Gamma \vdash \mathtt{t}_0.\mathtt{f}_\mathtt{i} : \mathtt{C}_\mathtt{i}} \text{ (FJ-FIELD)}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathtt{t}_0 : \mathtt{C}_0 \qquad \mathtt{mtype}(\mathtt{m}, \mathtt{C}_0) = \bar{\mathtt{D}} \to \mathtt{C} \\ \Gamma \vdash \bar{\mathtt{t}} : \bar{\mathtt{C}} \qquad \bar{\mathtt{C}} <: \bar{\mathtt{D}}\end{array}}{\Gamma \vdash \mathtt{t}_0.\mathtt{m}(\bar{\mathtt{t}}) : \mathtt{C}} \text{ (FJ-INVK)}$$

$$\frac{\mathtt{fields}(\mathtt{C}) = \bar{\mathtt{D}}\ \bar{\mathtt{f}} \qquad \Gamma \vdash \bar{\mathtt{t}} : \bar{\mathtt{C}} \qquad \bar{\mathtt{C}} <: \bar{\mathtt{D}}}{\Gamma \vdash \mathtt{new}\ \mathtt{C}(\bar{\mathtt{t}}) : \mathtt{C}} \text{ (FJ-NEW)}$$

$$\frac{\Gamma \vdash \mathtt{t}_0 : \mathtt{D} \qquad \mathtt{D} <: \mathtt{C}}{\Gamma \vdash (\mathtt{C})\mathtt{t}_0 : \mathtt{C}} \text{ (FJ-UCAST)}$$

$$\frac{\Gamma \vdash \mathtt{t}_0 : \mathtt{D} \qquad \mathtt{C} <: \mathtt{D} \qquad \mathtt{C} \neq \mathtt{D}}{\Gamma \vdash (\mathtt{C})\mathtt{t}_0 : \mathtt{C}} \text{ (FJ-DCAST)}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathtt{t}_0 : \mathtt{D} \qquad \mathtt{C} \not<: \mathtt{D} \qquad \mathtt{D} \not<: \mathtt{C} \\ s\mathit{tupid\ warning}\end{array}}{\Gamma \vdash (\mathtt{C})\mathtt{t}_0 : \mathtt{C}} \text{ (FJ-SCAST)}$$

$$\frac{\begin{array}{c}\bar{\mathtt{x}} : \bar{\mathtt{C}}, \mathtt{this} : \mathtt{C} \vdash \mathtt{t}_0 : \mathtt{E}_0 \qquad \mathtt{E}_0 <: \mathtt{C}_0 \\ \mathtt{CT}(\mathtt{C}) = \mathtt{class}\ \mathtt{C}\ \mathtt{extends}\ \mathtt{D}\ \{...\} \\ \mathtt{override}(\mathtt{m}, \mathtt{D}, \bar{\mathtt{C}} \to \mathtt{C}_0)\end{array}}{\mathtt{C}_0\ \mathtt{m}\ (\bar{\mathtt{C}}\ \bar{\mathtt{x}})\ \{\mathtt{return}\ \mathtt{t}_0;\}\ \mathtt{OK\ in}\ \mathtt{C}} \text{ (FJ-M-OK)}$$

$$\frac{\begin{array}{c}\mathtt{K} = \mathtt{C}\ (\bar{\mathtt{C}}\ \bar{\mathtt{f}})\{\mathtt{super}(\bar{\mathtt{f}});\ \mathtt{this}.\bar{\mathtt{f}} = \bar{\mathtt{f}}\} \\ \mathtt{fields}(\mathtt{D}) = \bar{\mathtt{D}}\ \bar{\mathtt{g}} \qquad \bar{\mathtt{M}}\ \mathtt{OK\ in}\ \mathtt{C}\end{array}}{\mathtt{class}\ \mathtt{C}\ \mathtt{extends}\ \mathtt{D}\ \{\bar{\mathtt{C}}\ \bar{\mathtt{f}}; \mathtt{K}\ \bar{\mathtt{M}}\}\ \mathtt{OK}} \text{ (FJ-C-OK)}$$

2 SWIN

2.1 Syntax

Π	$::=$	$\{\bar{\pi}\}$	SWIN program
π	$::=$	$(\bar{\mathtt{d}})\ [\mathtt{l} : \mathtt{C}_\mathtt{l}\ \to\ \mathtt{r} : \mathtt{C}_\mathtt{r}]$	rule
$\mathtt{d}$	$::=$	$\mathtt{x} : \mathtt{C}_1 \hookrightarrow \mathtt{C}_2$	variable declaration
$\mathtt{l}$	$::=$	$\mathtt{x.f} \mid \mathtt{new}\ \mathtt{C}(\bar{\mathtt{x}}) \mid \mathtt{x.m}(\bar{\mathtt{x}})$	code pattern
$\mathtt{r}$	$::=$	$\mathtt{t}$	FJ term

2.2 API definition

$$\mathtt{API}\ ::=\ \{\ \overline{\mathtt{CL}}\ \} \qquad \text{(API definition)}$$

2.3 Evaluation Rules

$$\frac{\mathtt{CL} = \mathtt{class}\ \mathtt{C_1}\ \mathtt{extends}\ \mathtt{C_2}\ \{\ \bar{\mathtt{C}}_\mathtt{i}\ \bar{\mathtt{f}}_\mathtt{i};\ \mathtt{K}\ \bar{\mathtt{M}}\ \}}{\Pi(\mathtt{CL}) = \mathtt{class}\ \Pi(\mathtt{C_1})\ \mathtt{extends}\ \Pi(\mathtt{C_2})\ \{\ \Pi(\bar{\mathtt{C}}_\mathtt{i})\ \bar{\mathtt{f}}_\mathtt{i};\ \Pi(\mathtt{K})\ \overline{\Pi(\mathtt{M})}\ \}}\ \text{(E-DECLARATION)}$$

$$\frac{\mathtt{K} = \mathtt{C_1}\ (\bar{\mathtt{C}}_2\ \bar{\mathtt{f}}_2)\ \{\mathtt{super}(\bar{\mathtt{f}}_3);\ \mathtt{this}.\bar{\mathtt{f}}_\mathtt{i} = \bar{\mathtt{f}}_\mathtt{j}\}}{\Pi(\mathtt{K}) = \Pi(\mathtt{C_1})\ (\Pi(\bar{\mathtt{C}}_2)\ \bar{\mathtt{f}}_2)\ \{\mathtt{super}(\bar{\mathtt{f}}_3);\ \mathtt{this}.\bar{\mathtt{f}}_\mathtt{i} = \bar{\mathtt{f}}_\mathtt{j}\}}\ \text{(E-CONSTRUCTOR)}$$

$$\frac{\mathtt{M} = \mathtt{C_1}\ \mathtt{m}(\bar{\mathtt{C}}_\mathtt{m}\ \bar{\mathtt{x}})\ \{\mathtt{return}\ \mathtt{t};\}}{\Pi(\mathtt{M}) = \Pi(\mathtt{C_1})\ \mathtt{m}(\Pi(\bar{\mathtt{C}}_\mathtt{m})\ \bar{\mathtt{x}})\ \{\mathtt{return}\ \Pi(\mathtt{t});\}}\ \text{(E-METHOD)}$$

$$\frac{\mathtt{C_0} \hookrightarrow \mathtt{C_1} \in \mathbf{TypeMapping}(\Pi)}{\Pi(\mathtt{C_0}) = \mathtt{C_1}}\ \text{(E-CLASS)}$$

$$\frac{\forall \mathtt{C}.\ \mathtt{C_0} \hookrightarrow \mathtt{C} \notin \mathbf{TypeMapping}(\Pi)}{\Pi(\mathtt{C_0}) = \mathtt{C_0}}\ \text{(E-ALTER-CLASS)}$$

$$\frac{}{\Pi(\mathtt{x}) = \mathtt{x}}\ \text{(E-T-VALUE)}$$

$$\frac{\begin{array}{c}(\mathtt{x} : \mathtt{C_1} \hookrightarrow \mathtt{C_2})[\ \mathtt{x.f} : \mathtt{C}\ \rightarrow\ \mathtt{r} : \mathtt{D}\ ] \in \Pi \qquad \mathbf{Type}(\mathtt{t}) <: \mathtt{C_1} \\ \nexists\ (\mathtt{x} : \mathtt{C_3} \hookrightarrow \mathtt{C_4})[\ \mathtt{x.f} : \mathtt{C}\ \rightarrow\ \mathtt{r} : \mathtt{D}\ ] \in \Pi.(\mathbf{Type}(\mathtt{t}) <: \mathtt{C_3} <: \mathtt{C_1} \land \mathtt{C_3} \neq \mathtt{C_1})\end{array}}{\Pi(\mathtt{t.f}) = [\ \mathtt{x} \mapsto \Pi(\mathtt{t})\ ]\mathtt{r}}\ \text{(E-T-FIELD)}$$

$$\frac{}{\Pi((\mathtt{C})\ \mathtt{t}) = (\Pi(\mathtt{C}))\ \Pi(\mathtt{t})}\ \text{(E-T-CAST)}$$

$$\frac{\begin{array}{c}(\bar{\mathtt{d}})[\ \mathtt{new}\ \mathtt{C_0}(\ \bar{\mathtt{x}}\ ) : \mathtt{C_1}\ \rightarrow\ \mathtt{r} : \mathtt{C_r}] \in \Pi \\ \{\ \bar{\mathtt{x}} : \overline{\mathtt{C_1} \hookrightarrow \mathtt{C_2}}\ \} \subseteq \bar{\mathtt{d}} \qquad \mathbf{Type}(\bar{\mathtt{t}}_\mathtt{u}) <: \bar{\mathtt{C}}_1\end{array}}{\Pi(\mathtt{new}\ \mathtt{C_0}(\bar{\mathtt{t}}_\mathtt{u})) = [\ \bar{\mathtt{x}} \mapsto \overline{\Pi(\mathtt{t_u})}\ ](\mathtt{r})}\ \text{(E-T-NEW)}$$

$$\frac{\begin{array}{c}(\bar{\mathtt{y}} : \overline{\mathtt{C_1} \hookrightarrow \mathtt{C_2}},\ \mathtt{x_0} : \mathtt{C_3} \hookrightarrow \mathtt{C_4})[\ \mathtt{x_0.m_0}(\ \bar{\mathtt{y}}\ ) : \mathtt{C}\ \rightarrow\ \mathtt{r} : \mathtt{D}] \in \Pi \\ \mathbf{Type}(\mathtt{t_0}) <: \mathtt{C_3} \qquad \mathbf{Type}(\bar{\mathtt{t}}_\mathtt{u}) <: \bar{\mathtt{C}}_1 \\ \nexists\ (\bar{\mathtt{y}} : \overline{\mathtt{C_1} \hookrightarrow \mathtt{C_2}},\ \mathtt{x_0} : \mathtt{C_5} \hookrightarrow \mathtt{C_6})[\ \mathtt{x_0.m_0}(\ \bar{\mathtt{y}}\ ) : \mathtt{C}\ \rightarrow\ \mathtt{r} : \mathtt{D}] \in \Pi.(\mathbf{Type}(\mathtt{t_0}) <: \mathtt{C_5} <: \mathtt{C_3} \land \mathtt{C_5} \neq \mathtt{C_3})\end{array}}{\Pi(\mathtt{t_0.m_0}(\bar{\mathtt{t}}_\mathtt{u})) = [\ \mathtt{x_0} \mapsto \Pi(\mathtt{t_0}),\ \bar{\mathtt{y}} \mapsto \overline{\Pi(\mathtt{t_u})}\ ](\mathtt{r})}\ \text{(E-T-INVOKE)}$$

$$\frac{\mathtt{no\ other\ inference\ rule\ can\ be\ applied}}{\Pi(\mathtt{new}\ \mathtt{C_0}(\bar{\mathtt{t}}_\mathtt{u})) = \mathtt{new}\ \mathtt{C_0}(\ \overline{\Pi(\mathtt{t_u})}\ )}\ \text{(E-ALTER-NEW)}$$

$$\frac{\mathtt{no\ other\ inference\ rule\ can\ be\ applied}}{\Pi(\mathtt{t_0.m_0}(\bar{\mathtt{t}}_\mathtt{u})) = \Pi(\mathtt{t_0}).\mathtt{m}(\ \overline{\Pi(\mathtt{t_u})}\ )}\ \text{(E-ALTER-INVOKE)}$$

$$\frac{\mathtt{no\ other\ inference\ rule\ can\ be\ applied}}{\Pi(\mathtt{t.f}) = \Pi(\mathtt{t}).\mathtt{f}}\ \text{(E-ALTER-FIELD)}$$

2.4 Auxiliary Functions

$$\textbf{TypeMapping}([(\ \bar{\mathtt{x}} : \overline{\mathtt{C_1} \hookrightarrow \mathtt{C_2}}\ )\ [\mathtt{l} : \mathtt{C_1}\ \rightarrow\ \mathtt{r} : \mathtt{C_r}]]) = \{\mathtt{C_1} \hookrightarrow \mathtt{C_r}\} \cup \{\ \overline{\mathtt{C_1} \hookrightarrow \mathtt{C_2}}\ \}$$

$$\textbf{TypeMapping}(\{\bar{\pi}\}) = \bigcup_{\pi} (\textbf{TypeMapping}(\pi)) \qquad \text{(Extract type migration information)}$$

$$\textbf{Decl}(\mathtt{class\ C\ extends\ D}\ \{...\}) = \mathtt{C} \qquad \text{(Extract the declared class name)}$$

2.5 Type Checking Rules

$$\frac{\{\ \bar{\mathtt{x}} : \bar{\mathtt{C}}\ \} \vdash_{\mathtt{l}}^{\mathtt{API_s}} \mathtt{l} : \mathtt{C_1} \qquad \{\ \bar{\mathtt{x}} : \bar{\mathtt{D}}\ \} \vdash_{\mathtt{FJ}}^{\mathtt{API_d}} \mathtt{r} : \mathtt{C_2}}{(\bar{\mathtt{x}} : \overline{\mathtt{C} \hookrightarrow \mathtt{D}})[\mathtt{l} : \mathtt{C_1} \rightarrow \mathtt{r} : \mathtt{C_2}]\ ok} \ \text{(T-}\pi\text{)}$$

$$\frac{\Gamma \vdash_{\mathtt{FJ}}^{\mathtt{API}} \mathtt{x} : \mathtt{C_0} \quad \mathtt{mtype}(\mathtt{m}, \mathtt{C_0}) = \bar{\mathtt{D}} \rightarrow \mathtt{C} \quad \Gamma \vdash_{\mathtt{FJ}}^{\mathtt{API}} \bar{\mathtt{y}} : \bar{\mathtt{D}}}{\Gamma \vdash_{\mathtt{l}}^{\mathtt{API}} \mathtt{x.m}(\bar{\mathtt{y}}) : \mathtt{C}} \ \text{(T-L1)}$$

$$\frac{\mathtt{fields}(\mathtt{C}) = \bar{\mathtt{D}}\ \bar{\mathtt{f}} \quad \Gamma \vdash_{\mathtt{FJ}}^{\mathtt{API}} \bar{\mathtt{x}} : \bar{\mathtt{D}}}{\Gamma \vdash_{\mathtt{l}}^{\mathtt{API}} \mathtt{new\ C}(\bar{\mathtt{x}}) : \mathtt{C}} \ \text{(T-L2)}$$

$$\frac{\mathtt{fields}(\mathtt{C}) = \bar{\mathtt{D}}\ \bar{\mathtt{f}} \quad \Gamma \vdash_{\mathtt{FJ}}^{\mathtt{API}} \mathtt{x} : \mathtt{C}}{\Gamma \vdash_{\mathtt{l}}^{\mathtt{API}} \mathtt{x.f_i} : \mathtt{D_i}} \ \text{(T-L3)}$$

$$\begin{aligned}
&\textbf{RuleOK}(\Pi) = \forall\ \pi.(\pi \in \Pi \Rightarrow \pi\ ok)\\
&\textbf{ConstrCover}(\Pi, \mathtt{API_s}, \mathtt{API_d}) =\\
&\quad \forall\ \mathtt{C_1}, \bar{\mathtt{C}}.(\mathtt{class\ C_1\ extends\ _}\ \{\mathtt{C_1}(\bar{\mathtt{C}}\ \bar{_})\ ...\ \} \in (\mathtt{API_s} - \mathtt{API_d})\\
&\qquad \Rightarrow\ \exists\ \mathtt{C_2}, \bar{\mathtt{C}}', \bar{\mathtt{x}}, \mathtt{r}.((\ \bar{\mathtt{x}} : \overline{\mathtt{C} \hookrightarrow \mathtt{C}'}\ )[\mathtt{new\ C_1}(\bar{\mathtt{x}}) : \mathtt{C_1} \rightarrow \mathtt{r} : \mathtt{C_2}] \in \Pi))\\
&\textbf{MethCover}(\Pi, \mathtt{API_s}, \mathtt{API_d}) =\\
&\quad \forall\ \mathtt{C_1}, \mathtt{C_2}, \mathtt{m}, \bar{\mathtt{C}}.(\mathtt{class\ C_1\ extends\ _}\ \{\ \mathtt{C_2\ m}(\ \bar{\mathtt{C}}\ \bar{_}\ )\{...\}\ ...\ \} \in (\mathtt{API_s} - \mathtt{API_d})\\
&\qquad \Rightarrow\ \exists\ \mathtt{x}, \bar{\mathtt{y}}, \mathtt{C}_1', \mathtt{C}_2', \bar{\mathtt{C}}', \mathtt{r}.((\mathtt{x} : \mathtt{C_1} \hookrightarrow \mathtt{C}_1',\ \ \bar{\mathtt{y}} : \overline{\mathtt{C} \hookrightarrow \mathtt{C}'}\ )[\mathtt{x.m}(\bar{\mathtt{y}}) : \mathtt{C_2} \rightarrow \mathtt{r} : \mathtt{C}_2'] \in \Pi))\\
&\textbf{FieldCover}(\Pi, \mathtt{API_s}, \mathtt{API_d}) =\\
&\quad \forall\ \mathtt{C_1}, \mathtt{C_2}, \mathtt{f}.(\mathtt{class\ C_1\ extends\ _}\ \{\mathtt{C_2\ f}; ...\} \in (\mathtt{API_s} - \mathtt{API_d})\\
&\qquad \Rightarrow\ \exists\ \mathtt{x}, \mathtt{C}_1', \mathtt{C}_2'.((\mathtt{x} : \mathtt{C_1} \hookrightarrow \mathtt{C}_1'\ )[\mathtt{x.f} : \mathtt{C_2} \rightarrow \mathtt{r} : \mathtt{C}_2'] \in \Pi))\\
&\textbf{MapChecking}(\Pi, \mathtt{API_s}, \mathtt{API_d}) =\\
&\quad \forall\ \mathtt{C}, \mathtt{D}.(\mathtt{C} \hookrightarrow \mathtt{D} \in \textbf{TypeMapping}(\Pi)\\
&\qquad \Rightarrow\ (\exists\ \mathtt{CL} \in \mathtt{API_s} \cap \mathtt{API_d}.(\textbf{Decl}(\mathtt{CL}) = \mathtt{C} \wedge \mathtt{D} = \mathtt{C}))\\
&\qquad\quad \vee (\exists\ \mathtt{CL} \in \mathtt{API_s} - \mathtt{API_d}.(\textbf{Decl}(\mathtt{CL}) = \mathtt{C})))\\
&\textbf{Subtyping}(\Pi, \mathtt{API_s}, \mathtt{API_d}) =\\
&\quad \forall\ \mathtt{C_i}, \mathtt{D_i}, \mathtt{C_j}, \mathtt{D_j}.(\mathtt{C_i} \hookrightarrow \mathtt{D_i}, \mathtt{C_j} \hookrightarrow \mathtt{D_j} \in \textbf{TypeMapping}(\Pi) \quad \Rightarrow \quad (\mathtt{C_i} <: \mathtt{C_j} \Rightarrow \mathtt{D_i} <: \mathtt{D_j}))
\end{aligned}$$

3 Metatheory

Lemma 1 (Typing Context)**.** *Given a SWIN program* Π *acting on* $\mathtt{API_s}$ *to* $\mathtt{API_d}$*, suppose the typing context for a term* $\mathtt{t}$ *is* $\Gamma_s = \bar{\mathtt{x}} : \bar{\mathtt{C}}$ *, then the typing context for* $\Pi(\mathtt{t})$ *is* $\Gamma_d = \bar{\mathtt{x}} : \overline{\Pi(\mathtt{C})}$*. (For simplicity, we use* $\Pi(\Gamma_s)$ *to represent* $\bar{\mathtt{x}} : \overline{\Pi(\mathtt{C})}$*, i.e.* Π *will act on all variable types in a typing context.)*

Proof. According to the FJ typing rules, the typing context will not change once it is created in the rule FJ-M-OK. For the typing context Γ, except the variable `this`, all other variables in the typing context are bounded in the definition of a method $\mathtt{M}$.

Induction on Γ. Suppose $\Gamma = \bar{\mathtt{y}} : \bar{\mathtt{D}}, \mathtt{x} : \mathtt{C} = \Gamma_1, \mathtt{x} : \mathtt{C}$, then $\Pi(\Gamma_1) = \bar{\mathtt{y}} : \overline{\Pi(\mathtt{D})}$.

There are two cases for $\mathtt{x} : \mathtt{C}$

- $\mathtt{x} = \mathtt{this}$ in Γ. The type C is a client defined class type, so $\mathtt{C} \notin \textbf{TypeMapping}(\Pi)$. According to the rule E-ALTER-CLASS, $\Pi(\mathtt{C}) = \mathtt{C}$, then we have $\Pi(\Gamma) = \Pi(\Gamma_0), \mathtt{x} : \Pi(\mathtt{C}) = \bar{\mathtt{y}} : \overline{\Pi(\mathtt{D})}, \mathtt{this} : \mathtt{C}$.

- x is an argument in method declaration. According to the rule E-METHOD, after transformation, the type of x in the definition is $\Pi(\mathtt{C})$, thus $\Gamma = \Pi(\Gamma_0), \mathtt{x} : \Pi(\mathtt{C}) = \bar{\mathtt{y}} : \overline{\Pi(\mathtt{D})}, \mathtt{x} : \Pi(\mathtt{C})$.

With these two cases proved, the lemma is proved. □

Lemma 2 (Subtyping)**.** *Suppose* Π *passes SWIN type checking rules, and it transforms an FJ program with* $\mathtt{API_s}$ *to a new program with* $\mathtt{API_d}$*, then:*
$\mathtt{C_1} <: \mathtt{C_2}$ *in old program* $\implies$ $\Pi(\mathtt{C_1}) <: \Pi(\mathtt{C_2})$ *in the transformed program.*

Proof. First, we suppose $\mathtt{C_1} <: \mathtt{C_2}$, in which $\mathtt{C_1} \neq \mathtt{C_2}$ and $\nexists\, \mathtt{C}$, s.t. $\mathtt{C_1} <: \mathtt{C'} <: \mathtt{C_2}$ and $\mathtt{C'} \neq \mathtt{C_1}, \mathtt{C'} \neq \mathtt{C_2}$. Consider the two possibilities for $\mathtt{C_1}$:

- case-1: class $\mathtt{C_1}$ is defined in client code.

  In this case, the declaration of $\mathtt{C_1}$ should be $\mathtt{CL} = \mathtt{class}\ \mathtt{C_1}\ \mathtt{extends}\ \mathtt{C_2}\ \{...\}$. According to the rule $\mathtt{E - DECLARATION}$, we have $\Pi(\mathtt{CL}) = \mathtt{class}\ \Pi(\mathtt{C_1})\ \mathtt{extends}\ \Pi(\mathtt{C_2})\ \{...\}$. Thus we have $\Pi(\mathtt{C_1}) <: \Pi(\mathtt{C_2})$.

- case-2: class $\mathtt{C_1}$ is defined in API.

  In this case we have $\mathtt{C_2}$ is also a API defined class according to the definition of API in FJ. According to the checking rule **ConstrCover**, these exists $\mathtt{C_1} \hookrightarrow \mathtt{D_1}, \mathtt{C_2} \hookrightarrow \mathtt{D_2} \in \textbf{TypeMapping}(\Pi)$. By the checking rule **Subtyping** and the fact that $\mathtt{C_1} <: \mathtt{C_2}$, we have $\mathtt{D_1} = \Pi(\mathtt{C_1}) <: \mathtt{D_2} = \Pi(\mathtt{C_2})$.

With this case proved, for any $\mathtt{C_1} <: \mathtt{C_2}$, it can be split into $\mathtt{C_1} <: \mathtt{C'} <: ... <: \mathtt{C_2}$. Applying the proof on each step by induction, the lemma is proved. □

Lemma 3 (Variable Substitution)**.** *Suppose that an FJ term* t *is well typed under context* $\Gamma = \Gamma_1, \{\bar{\mathtt{x}} : \bar{\mathtt{C}}_{\mathtt{x}}\}$*, i.e.* $\Gamma \vdash_{\mathtt{FJ}} \mathtt{t} : \mathtt{C_t}$*, then after substituting terms* $\bar{\mathtt{t}}_{\mathtt{u}}$ *for variables* $\bar{\mathtt{x}}$ *, with the property that* $\Gamma_1 \vdash_{\mathtt{FJ}} \bar{\mathtt{t}}_{\mathtt{u}} : \bar{\mathtt{C}}_{\mathtt{u}}$ *and* $\bar{\mathtt{C}}_{\mathtt{u}} <: \bar{\mathtt{C}}_{\mathtt{x}}$*,* t *can be typed to* $\mathtt{C_t}$ *or a sub-class of* $\mathtt{C_t}$*. Namely,*

$$\Gamma_1, \{\bar{\mathtt{x}} : \bar{\mathtt{C}}_{\mathtt{x}}\} \vdash_{\mathtt{FJ}} \mathtt{t} : \mathtt{C_t} \Longrightarrow \Gamma_1 \vdash_{\mathtt{FJ}} [\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_{\mathtt{u}}]\mathtt{t} : \mathtt{C'_t},\ \mathtt{C'_t} <: \mathtt{C_t}$$

Proof. By induction on the derivation on an FJ term t, there are five cases to discuss:

- case-1: $\mathtt{t} = \mathtt{x}$ and x is a variable.

  In this case, we substitute an FJ term $\mathtt{t_u}$ for x, where $\Gamma_1 \vdash_{\mathtt{FJ}} \mathtt{t_u} : \mathtt{C_u}$ and $\mathtt{C_u} <: \mathtt{C_t}$, then the substitution will be $[\mathtt{x} \mapsto \mathtt{t_u}]\mathtt{x} \Rightarrow \mathtt{t_u}$. As $\Gamma_1 \vdash_{\mathtt{FJ}} \mathtt{t_u} : \mathtt{C_u} <: \mathtt{C_t}$, we have $\Gamma_1 \vdash_{\mathtt{FJ}} [\mathtt{x} \mapsto \mathtt{t_u}]\mathtt{x} : \mathtt{C_u} <: \mathtt{C_t}$. Then we have the case proved.

- case-2: $\mathtt{t} = (\mathtt{C})\mathtt{t_1}$ and $\Gamma_1, \{\bar{\mathtt{x}} : \bar{\mathtt{C}}_{\mathtt{x}}\} \vdash_{\mathtt{FJ}} \mathtt{t_1} : \mathtt{C_1}$.

  In this case, by induction, we have $\Gamma_1 \vdash_{\mathtt{FJ}} [\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_{\mathtt{u}}]\mathtt{t_1} : \mathtt{C'_1}$ and $\mathtt{C'_1} <: \mathtt{C_1}$. For the term t, after substitution, we have $\Gamma_1 \vdash_{\mathtt{FJ}} (\mathtt{C})\mathtt{t_1} : \mathtt{C}$ and $\mathtt{C} <: \mathtt{C}$. Then we have the case proved.

- case-3: $\mathtt{t} = \mathtt{t_1.f}$ and $\Gamma_1, \{\bar{\mathtt{x}} : \bar{\mathtt{C}}_{\mathtt{x}}\} \vdash_{\mathtt{FJ}} \mathtt{t_1} : \mathtt{C_1}$.

  By induction, we have $\Gamma_1 \vdash_{\mathtt{FJ}} [\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_{\mathtt{u}}]\mathtt{t_1} : \mathtt{C'_1}$ and $\mathtt{C'_1} <: \mathtt{C_1}$. Then for field access, it will still access the same field f as it did before. Thus we have $\Gamma_1 \vdash_{\mathtt{FJ}} [\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_{\mathtt{u}}]\mathtt{t_1.f} : \mathtt{C_t}$ and $\mathtt{C_t} <: \mathtt{C_t}$, and the case is proved.

- case-4: $\mathtt{t} = \mathtt{new}\ \mathtt{C}_0(\bar{\mathtt{t}}_1)$ and $\Gamma_1, \{\bar{\mathtt{x}} : \bar{\mathtt{C}}_\mathtt{x}\} \vdash_{\mathtt{FJ}} \bar{\mathtt{t}}_1 : \bar{\mathtt{C}}_1$.

  The substitution $[\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\mathtt{t}$ equals to $\mathtt{new}\ \mathtt{C}_0([\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\bar{\mathtt{t}}_1)$.

  As the term $\mathtt{t}$ is well typed, we have:

$$\frac{\mathtt{fields}(\mathtt{C}_0) = \bar{\mathtt{D}}\ \bar{\mathtt{f}} \quad \Gamma \vdash_{\mathtt{FJ}} \bar{\mathtt{t}}_1 : \bar{\mathtt{C}}_1 \quad \bar{\mathtt{C}}_1 <: \bar{\mathtt{D}}}{\Gamma \vdash_{\mathtt{FJ}} \mathtt{new}\ \mathtt{C}(\bar{\mathtt{t}}_1) : \mathtt{C}_0}$$

  By induction, we have $\Gamma_1 \vdash_{\mathtt{FJ}} [\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\bar{\mathtt{t}}_1 : \bar{\mathtt{C}}'_1$ and $\bar{\mathtt{C}}'_1 <: \bar{\mathtt{C}}_1$, then we have the following derivation:

$$\frac{\mathtt{fields}(\mathtt{C}_0) = \bar{\mathtt{D}}\ \bar{\mathtt{f}} \quad \Gamma \vdash_{\mathtt{FJ}} [\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\bar{\mathtt{t}}_1 : \bar{\mathtt{C}}'_1 \quad \bar{\mathtt{C}}'_1 <: \bar{\mathtt{C}}_1 <: \bar{\mathtt{D}}}{\Gamma \vdash_{\mathtt{FJ}} \mathtt{new}\ \mathtt{C}([\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\bar{\mathtt{t}}_1) : \mathtt{C}_0}$$

  Thus we still have $\Gamma_1 \vdash_{\mathtt{FJ}} \mathtt{new}\ \mathtt{C}([\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\bar{\mathtt{t}}_1) : \mathtt{C}_0$ and $\mathtt{C}_0 <: \mathtt{C}_0$. This case is proved.

- case-5: $\mathtt{t} = \mathtt{t}_0.\mathtt{m}(\bar{\mathtt{t}}_1)$ and $\Gamma_1, \{\bar{\mathtt{x}} : \bar{\mathtt{C}}_\mathtt{x}\} \vdash_{\mathtt{FJ}} \bar{\mathtt{t}}_1 : \bar{\mathtt{C}}_1, \mathtt{t}_0 : \mathtt{C}_0$.

  In this case, the substitution $[\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\mathtt{t}$ equals to $([\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\mathtt{t}_0).\mathtt{m}([\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\bar{\mathtt{t}}_1)$.

  As the term $\mathtt{t}$ is well typed in FJ type system, we have:

$$\frac{\Gamma \vdash_{\mathtt{FJ}} \mathtt{t}_0 : \mathtt{C}_0 \quad \mathsf{mtype}(\mathtt{m}, \mathtt{C}_0) = \bar{\mathtt{D}} \to \mathtt{C} \quad \Gamma \vdash_{\mathtt{FJ}} \bar{\mathtt{t}}_1 : \bar{\mathtt{C}}_1 \quad \bar{\mathtt{C}}_1 <: \bar{\mathtt{D}}}{\Gamma \vdash_{\mathtt{FJ}} \mathtt{t}_0.\mathtt{m}(\bar{\mathtt{t}}_1) : \mathtt{C}}$$

  By induction, we have $\Gamma_1 \vdash_{\mathtt{FJ}} [\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\bar{\mathtt{t}}_1 : \mathtt{C}'_1, \mathtt{t}_0 : \mathtt{C}'_0$ and $\mathtt{C}'_0 <: \mathtt{C}_0, \bar{\mathtt{C}}'_1 <: \bar{\mathtt{C}}_1$.

  For the condition that $\mathtt{C}'_0 <: \mathtt{C}_0$, we have $\mathsf{mtype}(\mathtt{m}, \mathtt{C}'_0) = \mathsf{mtypem}(\mathtt{m}, \mathtt{C}_0)$ according to the rule METHOD-LOOKUP2.

  With these conditions, we have the following derivation for the new term after substitution:

$$\frac{\Gamma_1 \vdash_{\mathtt{FJ}} [\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\mathtt{t}_0 : \mathtt{C}'_0 \quad \mathsf{mtype}(\mathtt{m}, \mathtt{C}'_0) = \bar{\mathtt{D}} \to \mathtt{C} \quad \Gamma_1 \vdash_{\mathtt{FJ}} [\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\bar{\mathtt{t}}_1 : \bar{\mathtt{C}}'_\mathtt{t} \quad \bar{\mathtt{C}}'_1 <: \bar{\mathtt{C}}_1 <: \bar{\mathtt{D}}}{\Gamma_1 \vdash_{\mathtt{FJ}} [\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\mathtt{t}_0.\mathtt{m}([\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\bar{\mathtt{t}}_1) : \mathtt{C}}$$

  Thus we prove the case that: $\mathtt{Gamma}_1 \vdash_{\mathtt{FJ}} [\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\mathtt{t} : \mathtt{C}$ and $\mathtt{C} <: \mathtt{C}$.

With these five cases proved, we have the lemma proved. □

Lemma 4 (Term Formation)**.** *Given a well-typed SWIN program* Π*, if a term* $\mathtt{t}$ *in the original typing context can be typed to* $\mathtt{C}$*, then after transformation by* Π*, the term is well-typed and its type is a subtype of* $\Pi(\mathtt{C})$*. i.e.*

$$\Gamma_\mathtt{s} \vdash_{\mathtt{FJ}}^{\mathtt{API}_\mathtt{s}} \mathtt{t} : \mathtt{C} \Longrightarrow \Gamma_\mathtt{d} \vdash_{\mathtt{FJ}}^{\mathtt{API}_\mathtt{d}} \Pi(\mathtt{t}) : \mathtt{C}',\ \mathtt{where}\ \mathtt{C}' <: \Pi(\mathtt{C})$$

Proof. By induction on a derivation of a term $\mathtt{t}$. At each step of the induction, we assume that the desired property holds for all sub-derivations. We give our proof based on the last step of the derivation, which can only be one of the following five cases:

Before we move on to the cases analysis, we should note that according to Lemma 1, we have the relationship that $\Gamma_\mathtt{d} = \Pi(\Gamma_\mathtt{s})$.

- case-1: $\mathtt{t} = \mathtt{x}$ and $\Gamma_\mathtt{s} \vdash_{\mathtt{FJ}}^{\mathtt{API}_\mathtt{s}} \mathtt{x} : \mathtt{C}$.

  In this case, we have that $\Gamma_\mathtt{d} \vdash_{\mathtt{FJ}}^{\mathtt{API}_\mathtt{d}} \mathtt{x} : \Pi(\mathtt{C})$ according to Lemma 1. Then we have $\Gamma_\mathtt{d} \vdash_{\mathtt{FJ}}^{\mathtt{API}_\mathtt{d}} \mathtt{t} : \Pi(\mathtt{C})$ and $\Pi(\mathtt{C}) <: \Pi(\mathtt{C})$, and the case is proved.

- case-2: $\mathtt{t} = (\mathtt{C})\mathtt{t}_1$ and $\Gamma_\mathtt{s} \vdash_{\mathtt{FJ}}^{\mathtt{API}_\mathtt{s}} \mathtt{t}_1 : \mathtt{C}_1$.

  According to the rule E-T-CAST, $\Pi(\mathtt{t}) = \Pi(\mathtt{C})\ \Pi(\mathtt{t}_1)$.

  By induction, we have $\Gamma_\mathtt{d} \vdash_{\mathtt{FJ}}^{\mathtt{API}_\mathtt{d}} \mathtt{t}_1 : \mathtt{C}'_1$ and $\mathtt{C}'_1 <: \Pi(\mathtt{C}_1)$.

  By the rule FJ-*CAST (represent one of the three cast rules), we have $\Gamma_\mathtt{d} \vdash_{\mathtt{FJ}}^{\mathtt{API}_\mathtt{d}} (\Pi(\mathtt{C}))\ \Pi(\mathtt{t}_1) : \Pi(\mathtt{C})$, and $\Pi(\mathtt{C}) <: \Pi(\mathtt{C})$. Thus the case is proved.

- case-3: $\mathtt{t} = \mathtt{t}_1.\mathtt{f}$ and $\Gamma_\mathtt{s} \vdash_{\mathtt{FJ}}^{\mathtt{API_s}} \mathtt{t}_1 : \mathtt{C}_1$. In this case, there are further two subcases:
  - subcase-1: $\mathtt{C}_1$ is declared in $\mathtt{API_d}$, i.e. $\exists\ \mathtt{CL} \in \mathtt{API_s}$ and $\mathbf{Decl}(\mathtt{CL}) = \mathtt{C}_1$. Then we have $\Pi(\mathtt{C}_1)$ is declared in $\mathtt{API_d}$

    According to the checking rule **FieldCover**, we have a transformation rule

    $$\pi = (\mathtt{x} : \mathtt{C}_1 \hookrightarrow \mathtt{C}'_1)[\mathtt{x.f} : \mathtt{C} \hookrightarrow \mathtt{r} : \mathtt{C}'] \in \Pi$$

    to transform the field access expressions.

    By the evaluation rule E-T-FIELD, we have $\Pi(\mathtt{t}_1.\mathtt{f}) = [\mathtt{x} \mapsto \Pi(\mathtt{t}_1)]\mathtt{r}$. According to Lemma 3, we have $\Gamma_\mathtt{d} \vdash_{\mathtt{FJ}}^{\mathtt{API_d}} [\mathtt{x} \mapsto \Pi(\mathtt{t}_1)]\mathtt{r} : \mathtt{C}''$ and $\mathtt{C}'' <: \mathtt{C}'$.

    And by the definition of **TypeMapping**, there exists $\mathtt{C} \hookrightarrow \mathtt{C}'$ in $\mathbf{TypeMapping}(\Pi)$. Thus $\Pi(\mathtt{C}) = \mathtt{C}'$.

    With these properties, we have $\Gamma_\mathtt{d} \vdash_{\mathtt{FJ}}^{\mathtt{API_d}} \Gamma_\mathtt{d} \vdash_{\mathtt{FJ}}^{\mathtt{API_d}} [\mathtt{x} \mapsto \mathtt{t}_1]\mathtt{r} : \mathtt{C}''$ and $\mathtt{C}'' <: \mathtt{C}' = \Pi(\mathtt{C})$. And the subcase is proved.
  - subcase-2: $\mathtt{C}_1$ is defined in client class declarations.

    In this subcase, the rule will be evaluated by the rule E-ALTER-FIELD. By induction, we have $\Gamma \vdash_{\mathtt{FJ}}^{\mathtt{API_d}} \Pi(\mathtt{t}_1) : \mathtt{C}'_1$ and $\mathtt{C}'_1 <: \Pi(\mathtt{C}_1)$. Then according to the rule FJ-FIELD and the auxiliary function FIELD-LOOKUP and the evaluation rule E-DECLARATION, we have the following derivation tree:

    $$\frac{\Gamma \vdash_{\mathtt{FJ}}^{\mathtt{API_d}} \Pi(\mathtt{t}_1) : \mathtt{C}'_1 \qquad \dfrac{\mathtt{C}'_1 <: \Pi(\mathtt{C}_1) \quad \Pi(\mathtt{C})\ \mathtt{f} \in \mathsf{field}(\Pi(\mathtt{C}_1))}{\Pi(\mathtt{C})\ \mathtt{f} \in \mathsf{field}(\mathtt{C}'_1)}}{\Gamma \vdash_{\mathtt{FJ}}^{\mathtt{API_d}} \Pi(\mathtt{t}_1).\mathtt{f} : \Pi(\mathtt{C})}$$

    And of course, $\Pi(\mathtt{C}) <: \Pi(\mathtt{C})$, thus we have the subcase proved.

  With these two subcases proved, the case for field access is proved.
- case-4: $\mathtt{t} = \mathtt{new}\ \mathtt{C}(\bar{\mathtt{t}}_1)$ and $\Gamma \vdash_{\mathtt{FJ}}^{\mathtt{API_s}} \bar{\mathtt{t}}_1 : \bar{\mathtt{C}}_1$.

  In this case, we still have two subcases to discuss.
  - subcase-1: The class $\mathtt{C}$ is declared in $\mathtt{API_s}$, i.e. $\exists\ \mathtt{CL} \in \mathtt{API_s}.(\mathbf{Decl}(\mathtt{CL}) = \mathtt{C}_0)$, then $\Pi(\mathtt{C}_0)$ should be declared in $\mathtt{API_d}$.

    By the checking rule **ConstrCover**, there exists the following transformation rule to transform this term:

    $$\pi = (\bar{\mathtt{x}} : \overline{\mathtt{C}_2 \hookrightarrow \mathtt{C}'_2})[\mathtt{new}\ \mathtt{C}_0(\bar{\mathtt{x}}) : \mathtt{C} \hookrightarrow \mathtt{r} : \mathtt{C}']$$

    According to the rule E-T-NEW, we have:

    $$\frac{(\bar{\mathtt{x}} : \overline{\mathtt{C}_2 \hookrightarrow \mathtt{C}'_2})[\mathtt{new}\ \mathtt{C}_0(\bar{\mathtt{x}}) : \mathtt{C} \hookrightarrow \mathtt{r} : \mathtt{C}'] \in \Pi \qquad \dfrac{\mathbf{Type}(\bar{\mathtt{t}}_1) = \bar{\mathtt{C}}_1 \quad \bar{\mathtt{C}}_1 <: \bar{\mathtt{C}}_2}{\mathbf{Type}(\bar{\mathtt{t}}_1) <: \bar{\mathtt{C}}_2}}{\Pi(\mathtt{new}\ \mathtt{C}(\bar{\mathtt{t}}_1)) = [\bar{\mathtt{x}} \mapsto \overline{\Pi(\mathtt{t}_1)}]\mathtt{r}}$$

    By Lemma 3, we have $\Gamma_\mathtt{d} \vdash_{\mathtt{FJ}}^{\mathtt{API_d}} [\bar{\mathtt{x}} \mapsto \overline{\Pi(\mathtt{t})}_1]\mathtt{r} : \mathtt{C}''$ and $\mathtt{C}'' <: \mathtt{C}'$. And according to the definition of **TypeMapping** and the checking rule **Subtyping**, we have $\Pi(\mathtt{C}) = \mathtt{C}'$.

    Then we have $\Gamma_\mathtt{d} \vdash_{\mathtt{FJ}}^{\mathtt{API_d}} \Pi(\mathtt{t}) : \mathtt{C}''$ and $\mathtt{C}'' <: \Pi(\mathtt{C})$, and the subcase is proved.
  - subcase-2: The class $\mathtt{C}$ is declared in client code. Then the transformation of the term $\mathtt{t}$ should be evaluated according to the rule E-ALTER-NEW, thus we have $\Pi(\mathtt{t}) = \mathtt{new}\ \Pi(\mathtt{C})(\overline{\Pi(\mathtt{t}_1)})$.

    To finish the proof of the subcase, we need the following facts:

* According to the rule E-CONSTRUCTOR, the constructor definition is transformed to $\Pi(\mathtt{C})\ (\overline{\Pi(\mathtt{C}_2)}\ \bar{\mathtt{x}})\{...\}$. Thus $\mathsf{fields}(\Pi(C)) = \overline{\Pi(C_2)}$
* By induction, we have $\Gamma_{\mathtt{d}} \vdash_{\mathtt{FJ}}^{\mathtt{API_d}} \overline{\Pi(\mathtt{t}_1)} : \bar{\mathtt{C}}_1'$ and $\bar{\mathtt{C}}_1' <: \overline{\Pi(\mathtt{C}_1)}$.
* By the typing rule FJ-NEW, we have $\bar{\mathtt{C}}_1 <: \bar{\mathtt{C}}_2$. And by Lemma 2, we have $\overline{\Pi(\mathtt{C}_1)} <: \overline{\Pi(\mathtt{C}_2)}$.

With these facts, we have the following judgment:

$$\frac{\mathsf{fields}(\Pi(\mathtt{C})) = \overline{\Pi(\mathtt{C}_2)} \quad \Gamma_{\mathtt{d}} \vdash_{\mathtt{FJ}}^{\mathtt{API_d}} \overline{\Pi(\mathtt{t}_1)} : \bar{\mathtt{C}}_1' \quad \bar{\mathtt{C}}_1' <: \overline{\Pi(\mathtt{C}_1)} <: \overline{\Pi(\mathtt{C}_2)}}{\Gamma_{\mathtt{d}} \vdash_{\mathtt{FJ}}^{\mathtt{API_d}} \mathtt{new}\ \Pi(\mathtt{C})(\overline{\Pi(\mathtt{t}_1)}) : \Pi(\mathtt{C})}$$

And this proved the subcase.

With these two subcases proved, the case for object creation is proved.

* case-5: $\mathtt{t} = \mathtt{t}_1.\mathtt{m}(\bar{\mathtt{t}}_2)$ and $\Gamma_{\mathtt{s}} \vdash_{\mathtt{FJ}}^{\mathtt{API_s}} \mathtt{t}_1 : \mathtt{C}_1, \bar{\mathtt{t}}_2 : \bar{\mathtt{C}}_2$.

Similar to the previous one, there are also two subcases for this case.

– subcase-1: the class $\mathtt{C}_1$ is declared in $\mathtt{API_s}$. i.e. $\exists\, \mathtt{CL} \in \mathtt{API_s}.(\mathbf{Decl}(\mathtt{CL}) = \mathtt{C}_0)$.

In this subcase, the term $\mathtt{t}_1.\mathtt{m}(\bar{\mathtt{t}}_2)$ will be transformed by a rule according to E-T-INVOKE as the checking rule `MethCover` guarantees that there is a rule in Π to transform the method (At least a rule exists to transform the method `m` declared in a parent class of `C`).

Suppose the transformation rule is the following one (And this one is the closest rule to transform):

$$(\bar{\mathtt{y}} : \overline{\mathtt{C}_3 \hookrightarrow \mathtt{C}_3'}, \mathtt{x} : \mathtt{C}_4 \hookrightarrow \mathtt{C}_4')[\mathtt{x.m}(\bar{\mathtt{y}}) : \mathtt{C} \to \mathtt{r} : \mathtt{D}]$$

Then by the rule E-T-INVOKE, the transformation will be :

$$\frac{\begin{array}{c}(\bar{\mathtt{y}} : \overline{\mathtt{C}_3 \hookrightarrow \mathtt{C}_3'}, \mathtt{x} : \mathtt{C}_4 \hookrightarrow \mathtt{C}_4')[\mathtt{x.m}(\bar{\mathtt{y}}) : \mathtt{C} \to \mathtt{r} : \mathtt{D}] \in \Pi \quad \mathbf{Type}(\mathtt{t}_1) <: \mathtt{C}_4 \quad \mathbf{Type}(\bar{\mathtt{t}}_2) <: \bar{\mathtt{C}}_3 \\ \nexists(\bar{\mathtt{y}} : \overline{\mathtt{C}_3 \hookrightarrow \mathtt{C}_3'}, \mathtt{x} : \mathtt{C}_5 \hookrightarrow \mathtt{C}_5')[\mathtt{x.m}(\bar{\mathtt{y}}) : \mathtt{C} \to \mathtt{r} : \mathtt{D}] \in \Pi.(\mathbf{Type}(\mathtt{t}_0) <: \mathtt{C}_5 \wedge \mathtt{C}_5 \neq \mathtt{C}_4)\end{array}}{\Pi(\mathtt{t}_1.\mathtt{m}(\bar{\mathtt{t}}_2)) = [\mathtt{x} \mapsto \Pi(\mathtt{t}_1), \bar{\mathtt{y}} \mapsto \overline{\Pi(\mathtt{t}_2)}]\mathtt{r}}$$

By Lemma 3, we have $\Gamma_{\mathtt{d}} \vdash_{\mathtt{FJ}}^{\mathtt{API_d}} [\mathtt{x} \mapsto \Pi(\mathtt{t}_1), \bar{\mathtt{y}} \mapsto \overline{\Pi(\mathtt{t}_2)}]\mathtt{r} : \mathtt{C}'$ and $\mathtt{C}' <: \mathtt{D}$. Also, by the definition of **TypeMapping**, we have $\Pi(\mathtt{C}) = \mathtt{D}$. Thus we have $\Gamma \vdash_{\mathtt{FJ}}^{\mathtt{API_d}} \mathtt{t}_1.\mathtt{m}(\bar{\mathtt{t}}_2) : \mathtt{C}'$ and $\mathtt{C}' <: \Pi(\mathtt{C})$. And this subcase is proved.

– subcase-2: the class $\mathtt{C}_1$ is declared in client code. And in this case, the term `t` will be transformed by the rule E-ALTER-INVOKE:

$$\Pi(\mathtt{t}_1.\mathtt{m}(\bar{\mathtt{t}}_2)) = \Pi(\mathtt{t}_1).\mathtt{m}(\overline{\Pi(\mathtt{t}_2)})$$

To finish the proof of this case, we need the following points:

* By induction, we have $\Gamma_{\mathtt{d}} \vdash_{\mathtt{FJ}}^{\mathtt{API_d}} \Pi(\mathtt{t}_1) : \mathtt{C}_1'$ and $\mathtt{C}_1' <: \Pi(\mathtt{C}_1)$, $\Gamma_{\mathtt{d}} \vdash_{\mathtt{FJ}}^{\mathtt{API_d}} \overline{\Pi(\mathtt{t}_2)} : \bar{\mathtt{C}}_2'$ and $\bar{\mathtt{C}}_2' <: \overline{\Pi(\mathtt{C}_2)}$.
* According to the well-typedness of the original term in $\mathtt{API_s}$, we have the following derivation:

$$\frac{\Gamma_{\mathtt{s}} \vdash_{\mathtt{FJ}}^{\mathtt{API_s}} \mathtt{t}_1 : \mathtt{C}_1 \quad \mathsf{mtype}(\mathtt{m}, \mathtt{C}_1) = \bar{\mathtt{C}}_{\mathtt{u}} \to \mathtt{C} \quad \Gamma_{\mathtt{s}} \vdash_{\mathtt{FJ}}^{\mathtt{API_s}} \bar{\mathtt{t}}_2 : \bar{\mathtt{C}}_2 \quad \bar{\mathtt{C}}_2 <: \bar{\mathtt{C}}_{\mathtt{u}}}{\Gamma_{\mathtt{s}} \vdash_{\mathtt{FJ}}^{\mathtt{API_s}} \mathtt{t}_1.\mathtt{m}(\bar{\mathtt{t}}_2) : \mathtt{C}}$$

* According to the rule E-DECLARATION and E-METHOD, $\mathsf{mtype}(\mathtt{m}, \Pi(\mathtt{C}_1) = \Pi(\bar{\mathtt{C}}_{\mathtt{u}}) \to \Pi(\mathtt{C})$. By the definition of mtype, we have $\mathsf{mtype}(\mathtt{m}, \mathtt{C}_1') = \mathsf{mtype}(\mathtt{m}, \Pi(\mathtt{C}_1))$.

* By the checking rule **Subtyping**, we have $\overline{\Pi(\texttt{C}_2)} <: \overline{\Pi(\texttt{C}_\texttt{u})}$

With these facts, we can derive the type of the transformed term using the following tree:

$$\frac{\begin{array}{c}\Gamma_\texttt{d} \vdash_{\texttt{FJ}}^{\texttt{API}_\texttt{d}} \Pi(\texttt{t}_1) : \texttt{C}'_1 \quad \mathsf{mtype}(\texttt{m}, \texttt{C}'_1) = \mathsf{mtype}(\texttt{m}, \Pi(\texttt{C}_1)) = \overline{\Pi(\texttt{C}_\texttt{u})} \to \Pi(\texttt{C}) \\ \Gamma_\texttt{d} \vdash_{\texttt{FJ}}^{\texttt{API}_\texttt{d}} \overline{\Pi(\texttt{t}_2)} : \bar{\texttt{C}}'_2 \quad \bar{\texttt{C}}'_2 <: \overline{\Pi(\texttt{C}_2)} <: \overline{\Pi(\texttt{C}_\texttt{u})}\end{array}}{\Gamma_\texttt{d} \vdash_{\texttt{FJ}}^{\texttt{API}_\texttt{d}} \Pi(\texttt{t}_1).\texttt{m}(\overline{\Pi(\texttt{t}_2)}) : \Pi(\texttt{C})}$$

And thus we have $\Gamma_\texttt{d} \vdash_{\texttt{FJ}}^{\texttt{API}_\texttt{d}} \Pi(\texttt{t}) : \Pi(\texttt{C})$ in this subcase.

With these two subcases proved, we have the lemma holds for the case 5.

With these five cases for a term proved, by induction, the lemma holds for any FJ term. □

Lemma 5 (Method Formation)**.** *An FJ method declaration is well formed after transformed by a well-typed SWIN program. i.e. For any* $\texttt{M}$*,*

$$\Pi(\texttt{M}) = \Pi(\texttt{C}_1)\ \texttt{m}(\Pi(\bar{\texttt{C}}_\texttt{m})\ \bar{\texttt{x}})\ \{\texttt{return}\ \Pi(\texttt{t});\}$$

is well-formed with new API if Π *is well typed.*

Proof. According to Lemma 4, we have $\{\bar{\texttt{x}} : \overline{\Pi(\texttt{C}_\texttt{m})}, \texttt{this} : \Pi(\texttt{C}_1)\} \vdash_{\texttt{FJ}}^{\texttt{API}_\texttt{d}} \texttt{t} : \texttt{C}'_1$ and $\texttt{C}'_1 <: \Pi(\texttt{C}_1)$.

Suppose $\texttt{CT}(\texttt{C}) = \texttt{class C extends D}\ \{\ldots\}$, according to the rule E-METHOD and E-DECLRATION, $\mathsf{override}(\texttt{m}, \Pi(\texttt{D}), \overline{\Pi(\texttt{C})} \to \texttt{C}_1)$.

Then the formation of the transformed term is proved by the FJ-M-OK derivation on these judgments:

$$\frac{\begin{array}{c}\{\bar{\texttt{x}} : \overline{\Pi(\texttt{C}_\texttt{m})}, \texttt{this} : \Pi(\texttt{C}_1)\} \vdash_{\texttt{FJ}}^{\texttt{API}_\texttt{d}} \texttt{t} : \texttt{C}'_1 \qquad \texttt{C}'_1 <: \Pi(\texttt{C}_1) \\ \texttt{CT}(\Pi(\texttt{C})) = \texttt{class}\ \Pi(\texttt{C}_1)\ \texttt{extends}\ \Pi(\texttt{D})\ \{\ldots\} \\ \mathsf{override}(\texttt{m}, \Pi(\texttt{D}), \overline{\Pi(\texttt{C})} \to \texttt{C}_1)\end{array}}{\Pi(\texttt{C}_1)\ \texttt{m}\ (\bar{\texttt{C}}\ \bar{\texttt{x}})\ \{\texttt{return}\ \texttt{t}_0;\}\ \texttt{OK in}\ \Pi(\texttt{C})} \ \text{(FJ-M-OK)}$$

□

Theorem 1 (Type-Safety)**.** *Any FJ program is well-typed after a transformation by a well-typed SWIN program* Π*. i.e. For any* $\texttt{CL}$*,*

$$\Pi(\texttt{CL}) = \texttt{class}\ \Pi(\texttt{C}_1)\ \texttt{extends}\ \Pi(\texttt{C}_2)\ \{\ \Pi(\bar{\texttt{C}}_\texttt{i})\ \bar{\texttt{f}}_\texttt{i};\ \Pi(\texttt{K})\ \overline{\Pi(\texttt{M})}\ \}$$

is well-typed with new API if Π *is well-typed.*

Proof. By Lemma 5, we have all method declarations well formed. And by the rule E-CONSTRUCTOR, we have the following derivation:

$$\frac{\Pi(\texttt{K}) = \Pi(\texttt{C})\ (\overline{\Pi(\texttt{C})}\ \bar{\texttt{f}})\{\ldots\} \quad \mathsf{fields}(\Pi(\texttt{D})) = \overline{\Pi(\texttt{D})}\ \bar{\texttt{g}} \quad \overline{\Pi(\texttt{M})}\ ok}{\texttt{class}\ \Pi(\texttt{C})\ \texttt{extends}\ \Pi(\texttt{D})\ \{\overline{\Pi(\texttt{C})}\ \bar{\texttt{f}}; \Pi(\texttt{K})\ \overline{\Pi(\texttt{M})}\}\ ok}$$

With this proved, we have the theorem proved, i.e a well-typed SWIN program can transform any FJ program correctly. □