# Adding a Checker

## Step 1: Register the New Checker

Edit the file ~/infer/src/checkers/registerCheckers.ml and add your new checker signature call to the Java checkers list. For example:

let java\_checkers =

let l =

[

…

MyCheckerClass.callback\_my\_checker\_function, checkers\_enabled;

…

] in

…

## Step 2:

Under the directory ~/infer/src/checkers/ add a new .ml (module, similar to a class file in C) and .mli (module interface, similar to a header file in C) files for you checker. For example:

### Step 2.1: The Checker Module (myCheckerClass.ml)

open! IStd

module P = Printf

(\* The report\_error method with arguments: activity\_typ, fld, fld\_typ, pname, pdesc

Defines how to report the error, called when an error is found

\*)

let report\_error activity\_typ fld fld\_typ pname pdesc =

let checker\_exception = "CHECKERS\_ACTIVITY\_RETAINS\_STATIC\_VIEW" in

let description = Localise.desc\_my\_checker activity\_typ fld fld\_typ pname in

let exn = Exceptions.Checkers (checker\_exception, description) in

let loc = Procdesc.get\_loc pdesc in

Reporting.log\_error pname ~loc exn

(\* checker definition \*)

let callback\_my\_checker\_function\_java

pname\_java { Callbacks.proc\_desc; tenv } =

(\* checker macro: check if the name of the analyzed function is "onDestroy" \*)

let is\_on\_destroy = String.equal (Procname.java\_get\_method pname\_java) "onDestroy" in

(\* checker macro: checks if the field [tname] is a view \*)

let fld\_typ\_is\_view = function

| Typ.Tptr (Tstruct tname, \_) -> AndroidFramework.is\_view tenv tname

| \_ -> false in

(\* is [fldname] a View type declared by [class\_typename]? \*)

let is\_declared\_view\_typ class\_typename (fldname, fld\_typ, \_) =

let fld\_classname = Typename.Java.from\_string (Ident.java\_fieldname\_get\_class fldname) in

Typename.equal fld\_classname class\_typename && fld\_typ\_is\_view fld\_typ in

if is\_on\_destroy then (\* checks for a specific analyzed function\*)

(\* Define class name variable\*)

let class\_typename =

Typename.Java.from\_string (Procname.java\_get\_class\_name pname\_java) in

(\* get all the fields in the class if it is an activity\*)

match Tenv.lookup tenv class\_typename with

| Some { fields } when AndroidFramework.is\_activity tenv class\_typename -> (\*run the following if fields are found in an activity \*)

(\* filter the declared views from all the fields \*)

let declared\_view\_fields =

List.filter ~f:(is\_declared\_view\_typ class\_typename) fields in

(\* get all the nullified fields \*)

let fields\_nullified = PatternMatch.get\_fields\_nullified proc\_desc in

(\* report if a field is declared by C, but not nulled out in C.onDestroyView \*)

(\* iterate over all declared views \*)

List.iter

~f:(fun (fname, fld\_typ, \_) ->

(\* check if the declared field in not nullified \*)

if not (Ident.FieldSet.mem fname fields\_nullified) then

(\* report an error! found a declared view field which is not nullified \*)

report\_error (Tstruct class\_typename) fname fld\_typ (Procname.Java pname\_java) proc\_desc

)

declared\_view\_fields

| \_ -> () (\* do nothing if no fileds in activity found \*)

(\* main \*)

let callback\_my\_checker\_function ({ Callbacks.proc\_name } as args) =

match proc\_name with

| Procname.Java pname\_java -> (\* if the analyzer function is a java function \*)

callback\_my\_checker\_function\_java pname\_java args

| \_ -> () (\* do nothing if this is not a java function \*)

See <http://fbinfer.com/docs/adding-checkers.html> for more information.

### Step 2.1: The Checker Module Interface (myCheckerClass.mli)

val callback\_callback\_my\_checker\_function: Callbacks.proc\_callback\_t

## Step 3: Defining the Printed Output

### Step 3.1:

Edit the file ~/infer/src/IR/Localise.ml to add support for you need checker. For example:

…

let desc\_my\_checker activity\_typ fieldname fld\_typ pname : error\_desc =

let problem =

Printf.sprintf "Activity %s does not handle field %s (type %s) in %s correctly."

(format\_typ activity\_typ)

(format\_field fieldname)

(format\_typ fld\_typ)

(format\_method pname) in

let consequences =

"Some description of why this is a problem." in

let advice =

"Some suggestions on how to solve the problem." in

{ no\_desc with descriptions = [problem; consequences; advice] }

…

### Step 3.2:

Also edit the file ~/infer/src/IR/Localise.mli to add support for you need checker. For example:

…

val desc\_my\_checker :

Typ.t -> Ident.fieldname -> Typ.t -> Procname.t -> error\_desc

…

## Step 4: Build Infer

Run: ./build-infer.sh java