Exercise 5.1: Name Binding and Method Calls

The following source code is written in an object-oriented language.

class T is
  ...
  method M (X : T) is begin Print ("T"); end M;
end class T;

class NT inherits T is
  ...
  method M (X : T) is begin Print ("NT"); end M; -- redefinition!
  method M2 (X : NT) is begin ... end M2;
end NT;

OT : T;
ONT : NT := new NT;
OT := new NT;
...

OT.M (OT); -- (1)
OT.M2 (ONT); -- (2)
ONT.M (OT); -- (3)
ONT.M2 (ONT); -- (4)

a. Assume that the language uses dynamic name binding for method calls (like Smalltalk).
   Are the calls at (1) to (4) legal calls? If so, which output will be produced by each call (assume that the calls are alternative implementations, so they do not influence each other)? If not, what kind of error diagnostic do you expect?

b. In the following, the language uses static name binding for method calls (like C++, Java, Eiffel). Repeat part a. Which results are the same? Which differ? Please state the reason for the observed behaviour.

c. What are the advantages of static name binding for method calls?

d. General Question: Explain how a method name is bound to some method declaration and how the correct body of the method is found for languages using static name binding.
Exercise 5.2: Indirect Function Calls

a. What problem can arise with the lifetime of variables in the context of indirect function calls?

b. Why is this problem with direct function calls cannot emerge in conventional programming languages?

Exercise 5.3: Overloading

a. Give examples of overloading in Programming Languages.

b. Explain the difference between overloading and overriding?

c. Does overloading cause additional overhead during runtime?

d. Is there a method to statically resolve Overloading in the compiler.