Compilers

Pluggable Type Checkers

Salahaddin University
College of Engineering
Software Engineering Department
2011-2012

Amanj Sherwany
http://www.amanj.me/wiki/doku.php?id=teaching:su:compilers

Type Systems

- Traditionally, statically typed programming languages incorporate a built-in static type system.
- This system is typically mandatory:
  - Every legal program must successfully typecheck according to the rules of the type system.
- Mainstream programming languages incorporate mandatory type systems, like Java, C and C++.

Type Systems, *Cont'd*

- These type systems are widely perceived to enhance software reliability and security, by mechanically proving program properties.
- Paradoxically, such mandatory type systems inhibit software reliability and security, because they contribute to system complexity and brittleness.

Type Systems, *Cont'd*

- The advantages of mandatory typing are widely recognized:
  - Types provide a form of machine-checkable documentations.
  - Types provide a conceptual framework for the programmer, that is extremely useful for program design, maintenance and understanding.
  - Types support early error detection.
  - Languages with mandatory types can be optimized based on type information, yielding significant performance advantage.

Type Systems, *Cont'd*

- Mandatory type systems constrain the expressiveness of the language.
- Mandatory type systems greatly reduce the number of legal programs that can be expressed in a language.

Pluggable Type Checkers

- Or, optional type systems
- First proposed by Bracha
- Allow static checking of different program properties at different stages.
- In Bracha's terms:
  - Have no effect on the run-time semantics of the programming language.
  - Do not mandate type annotations in the syntax.

Demo

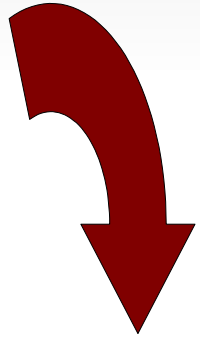
DEMO

How Pluggable Type Checkers Work

```
import checkers.nullness.quals.*;
public class Test{
    void print(@NonNull String str){
        System.out.println(str);
    }
}
```

How Pluggable Type Checkers Work

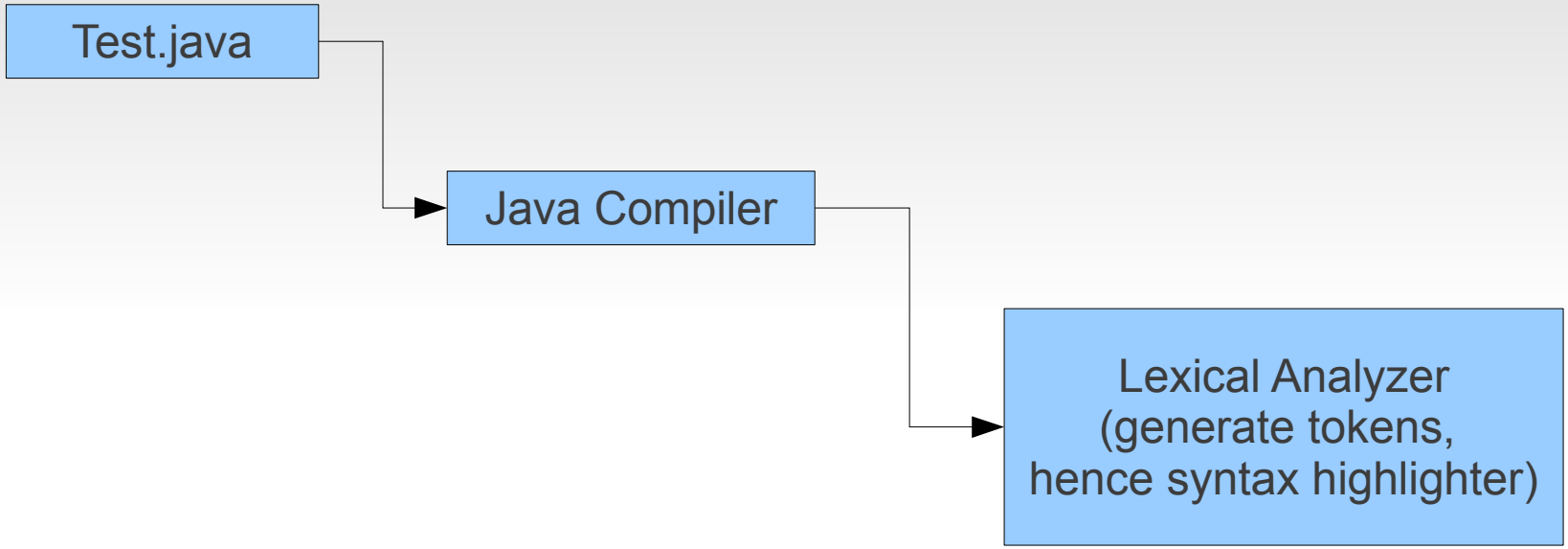
```
import checkers.nullness.quals.*;
public class Test{
    void print(@NonNull String str){
        System.out.println(str);
    }
}
```



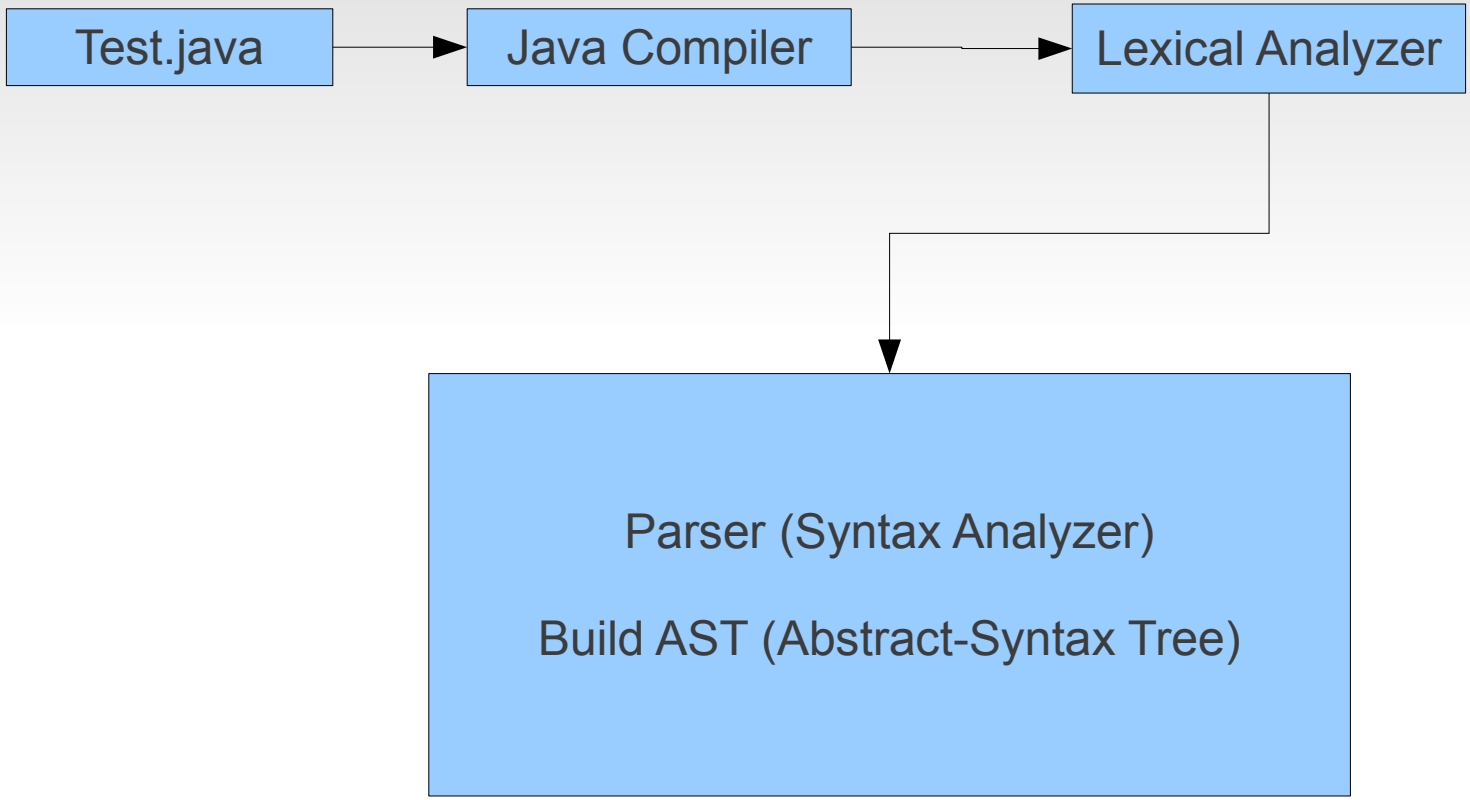
Compile it

```
javac -processor checkers.nullness.NullnessChecker Test.java
```

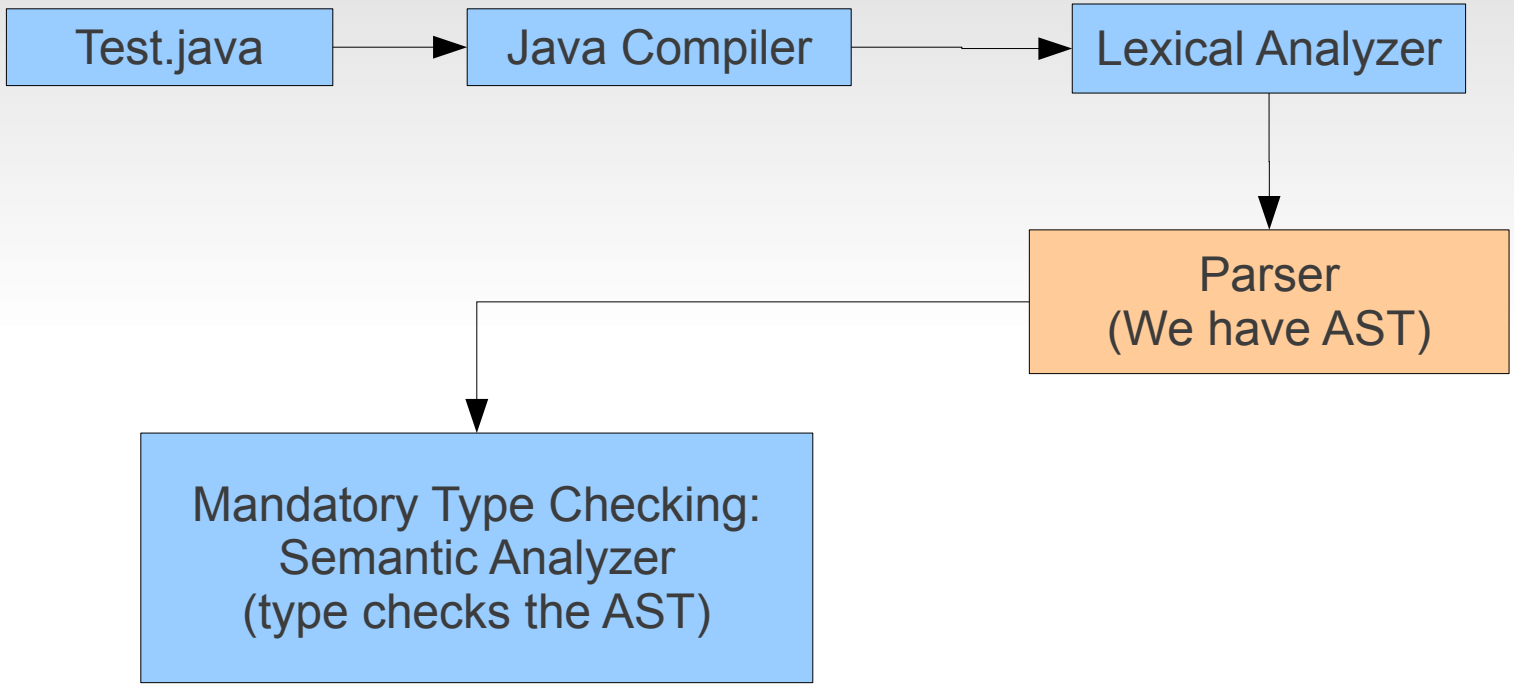
How Pluggable Type Checkers Work



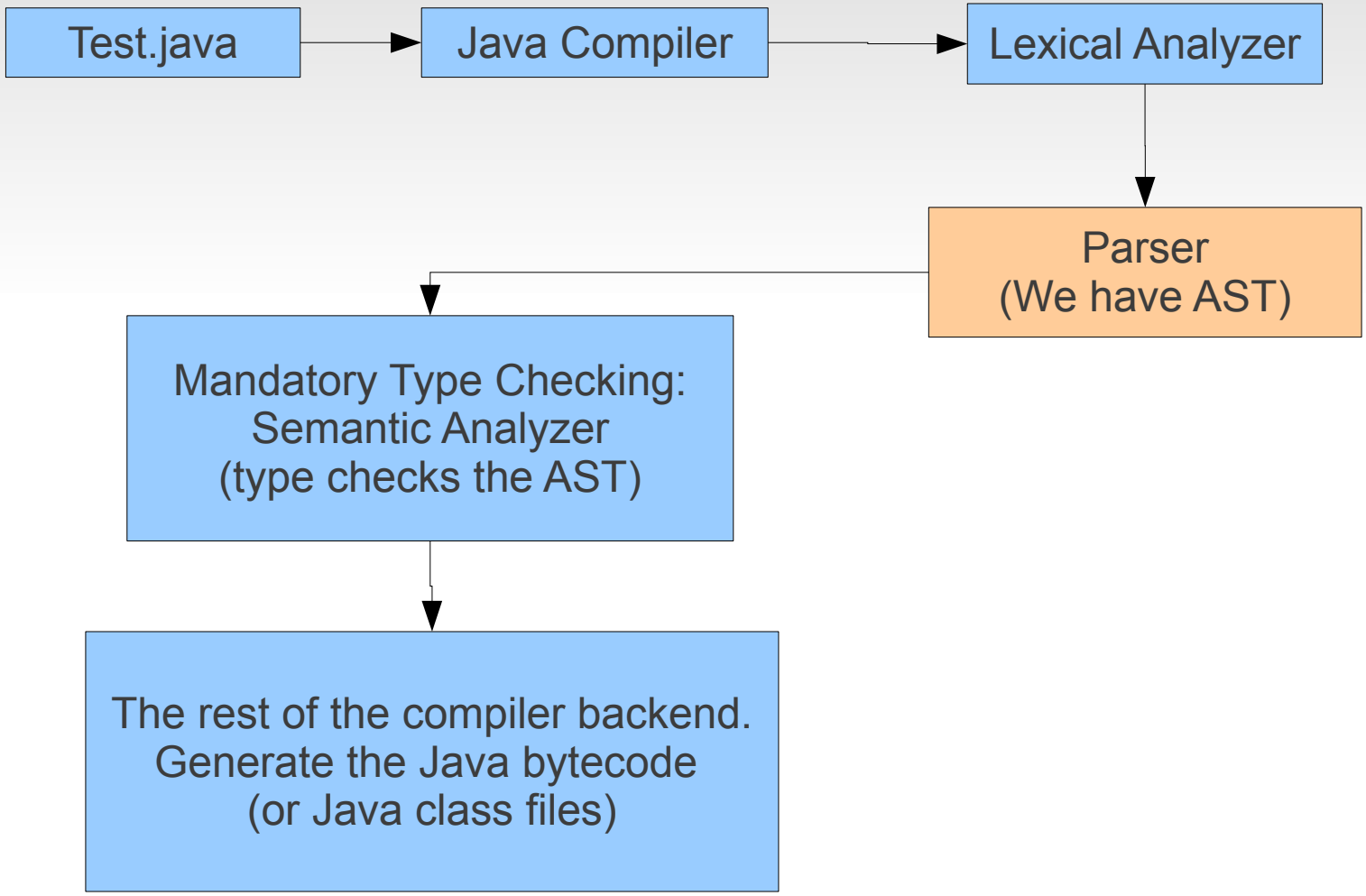
How Pluggable Type Checkers Work



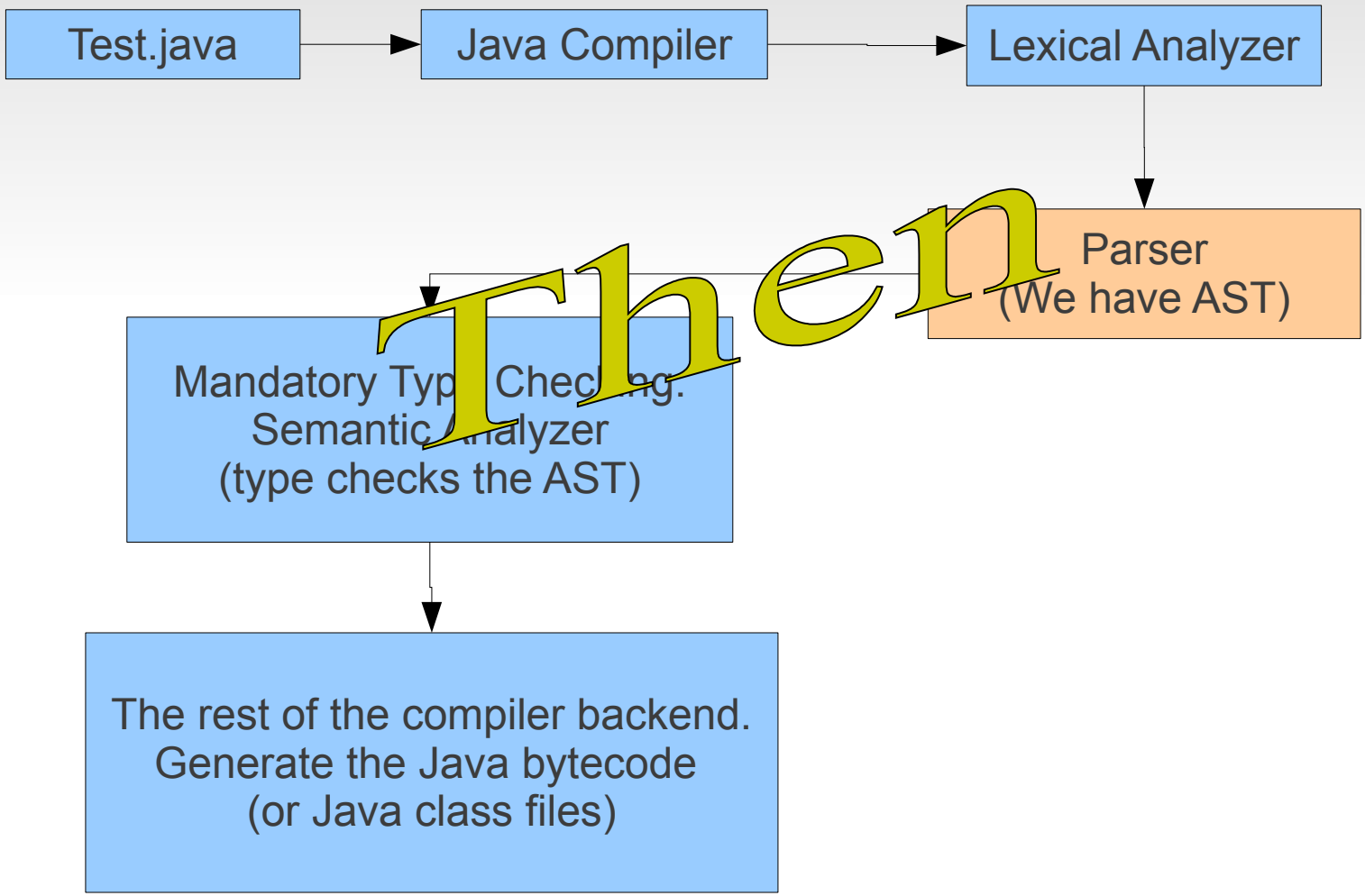
How Pluggable Type Checkers Work



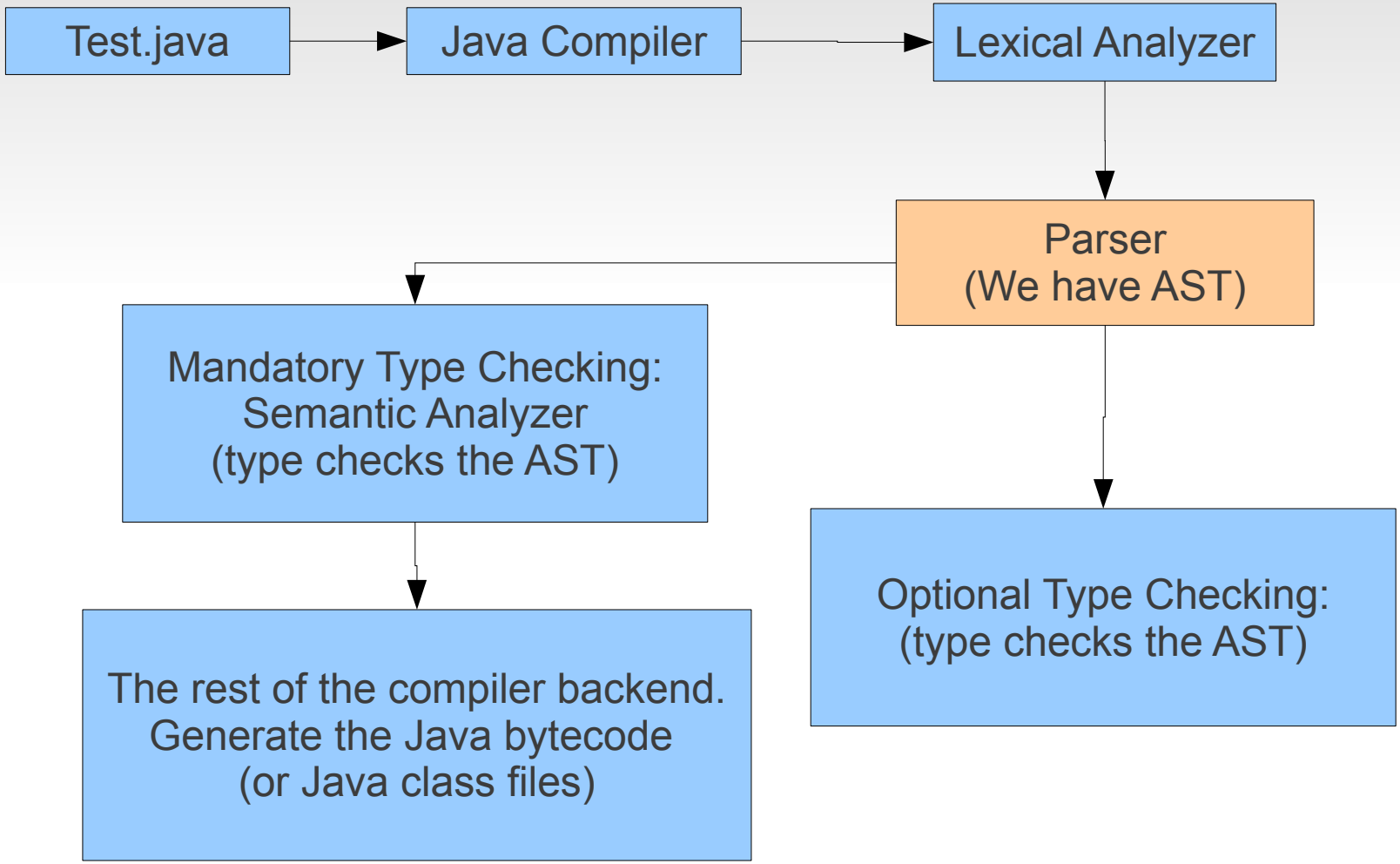
How Pluggable Type Checkers Work



How Pluggable Type Checkers Work



How Pluggable Type Checkers Work



Questions?

Thank You