Starting Small

Ron Pressler

Starting Small

- Java is the leading language for big, long-lasting, server-side programs because it's great at scaling *up*
- Java has lost ground in education and in smaller software because it's not so great at scaling *down*
- Every large project starts out small
- Every expert starts out a beginner
- Incumbents are always disrupted from *below*

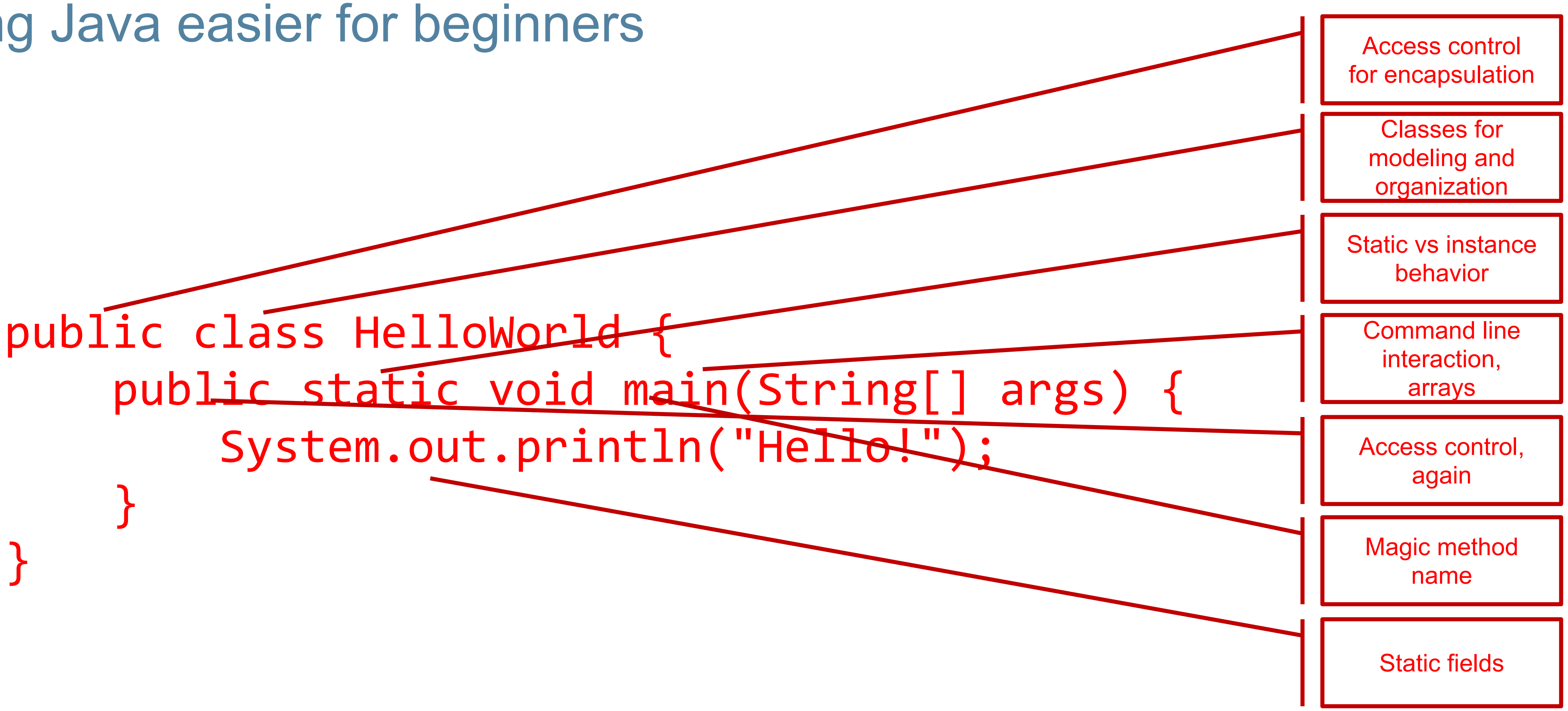
Starting Small

- Reduce effort to learning for beginners, as well as for starting a project for experts
- Do not introduce a separate “beginners’ dialect” of Java
- Do not introduce a special tooling workflow for beginners
- Changes must be a *natural*, consistent evolution of the Java language and tooling
- A series of independent JEPs covering the language, existing tools, and new tools

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello!");
    }
}
```

Paving the on-ramp

Making Java easier for beginners



```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello!");
    }
}
```

Access control for encapsulation
Classes for modeling and organization
Static vs instance behavior
Command line interaction, arrays
Access control, again
Magic method name
Static fields

Programming in the large and in the small

- **Programming in the large** composing components through interfaces
- **Programming in the small** component internals
- What's large and small is relative
    - Module — an unnamed one is provided implicitly
    - Package — an unnamed one is provided implicitly
    - Class — an unnamed one will now be provided implicitly
- Access modifiers are a mechanism for programming in the large

- [JEP 512: Compact Source Files and Instance Main Methods]()
- [JEP 511: Module Import Declarations]()

```
void main() {
    IO.println(“Hello!");
}
```

main

- Static or instance
- With or without String[] args
- With or without an explicit class declaration

Starting Small

- Even big projects done by experts start small — tinkering and exploration
- JShell (JEP 222, integrated in JDK 9) — tinkering with statements
- Launch Single-File Source-Code Programs (JEP 330, JDK 11) — tinkering with one file
- Once we have more than one file we configure a build tool

JEP 458: Launch Multi-File Source-Code Programs

- Let programmers choose when they want to set up a build configuration
- We will allow launching *multi-file* source code programs, without a compilation step

```
// - Prog.java
class Prog {
    public static void main(String[] args) { Helper.run(); }
}

// - Helper.java
class Helper {
    static void run() { System.out.println("Hello!"); }
}

// - lib1.jar
// - lib2.jar
```

```
java -cp '*' Prog.java
```

Launch Multi-File Source-Code Programs

- Works when source files span multiple packages
- Works when source files span a single module
- Works even with dynamically-loaded classes (`Class.forName`)
    - A custom class-loader compiles sources on-demand

Starting Small

- What about downloading and using libraries?
- What about choosing and learning a build tool?

Integrity by Default

A Gradual Yet Resolute Path Forward

- JEP draft: Integrity and Strong Encapsulation
- JEP 451: Prepare to Disallow the Dynamic Loading of Agents
- JEP 472: Prepare to Restrict the Use of JNI
- JEP 498: Warn upon Use of Memory-Access Methods in sun.misc.Unsafe
  - JEP 471: Deprecate the Memory-Access Methods in sun.misc.Unsafe for Removal
- JEP 500: Prepare to Make Final Mean Final