ripgrep crates/grep/src/lib.rs: Code Companion

Reference code for the The Facade Pattern lecture. Sections correspond to the lecture document.

Section 1: The Module Documentation Block

```
/*!
ripgrep, as a library.

This library is intended to provide a high level facade to the crates that
make up ripgrep's core searching routines. However, there is no high level
documentation available yet guiding users on how to fit all of the pieces
together.

Every public API item in the constituent crates is documented, but examples
are sparse.

A cookbook and a guide are planned.
*/
```

The `/*!` syntax creates an "inner" doc comment that documents the enclosing item (the crate itself) rather than the following item. This becomes the crate's root documentation page. Compare with `///` which documents the item that follows it.

Section 2: Public Re-exports as Architecture

```
// Each `pub extern crate` makes an external crate available as a submodule
// The `as` clause renames the crate for cleaner access

pub extern crate grep_cli as cli;         // grep::cli
pub extern crate grep_matcher as matcher; // grep::matcher
pub extern crate grep_printer as printer; // grep::printer
pub extern crate grep_regex as regex;     // grep::regex
pub extern crate grep_searcher as searcher; // grep::searcher
```

The renaming strips the `grep_` prefix, transforming ecosystem-safe crate names into ergonomic module names. Users write `grep::searcher::Searcher` instead of `grep_searcher::Searcher`.

Section 3: Conditional Compilation with Feature Flags

```rust
// This re-export only exists when the "pcre2" feature is enabled
// in Cargo.toml: grep = { version = "...", features = ["pcre2"] }
#[cfg(feature = "pcre2")]
pub extern crate grep_pcre2 as pcre2;
```

The `#[cfg(...)]` attribute performs compile-time conditional inclusion. When `pcre2` feature is disabled, this line is completely removed from compilation—no dead code, no unused dependency.

Section 4: The Crate Dependency Graph

```rust
// Core abstraction - defines what a "matcher" must do
pub extern crate grep_matcher as matcher;

// Implementations of the Matcher trait
pub extern crate grep_regex as regex;      // Pure Rust regex engine
#[cfg(feature = "pcre2")]
pub extern crate grep_pcre2 as pcre2;      // C library wrapper

// Orchestration and output
pub extern crate grep_searcher as searcher; // File reading + matching
pub extern crate grep_printer as printer;   // Result formatting

// CLI utilities
pub extern crate grep_cli as cli;           // Terminal handling, etc.
```

The dependency flow: `matcher` defines traits → `regex` / `pcre2` implement them → `searcher` uses matchers → `printer` formats results → `cli` handles terminal concerns.

Section 5: The Builder Pattern Connection

```
// Through the facade, users access builders from various crates:

// From grep::searcher (grep_searcher crate)
// SearcherBuilder::new().line_terminator(b'\n').build()

// From grep::printer (grep_printer crate)
// StandardBuilder::new().color_specs(specs).build(writer)

// From grep::regex (grep_regex crate)
// RegexMatcherBuilder::new().case_insensitive(true).build(pattern)

// The facade makes these discoverable under one namespace
pub extern crate grep_searcher as searcher;
pub extern crate grep_printer as printer;
pub extern crate grep_regex as regex;
```

Each builder follows Rust conventions: `new()` → chainable configuration methods → `build()` to produce the final type. The facade groups these related APIs.

Section 6: Why Not Just One Big Crate?

```
// Users who only need regex matching can depend on just:
// [dependencies]
// grep_matcher = "..."
// grep_regex = "..."

// Users who want everything use the facade:
// [dependencies]
// grep = "..."

// The facade provides the unified interface:
pub extern crate grep_cli as cli;
pub extern crate grep_matcher as matcher;
#[cfg(feature = "pcre2")]
pub extern crate grep_pcre2 as pcre2;
pub extern crate grep_printer as printer;
pub extern crate grep_regex as regex;
pub extern crate grep_searcher as searcher;
```

Separate crates enable parallel compilation, independent versioning, and selective dependency. The facade provides convenience without sacrificing these benefits.

Quick Reference

Re-export Syntax Comparison

Syntax	Effect	Use Case
`pub extern crate foo`	Re-exports crate as submodule	Facade patterns
`pub extern crate foo as bar`	Re-exports with rename	Cleaner API names
`pub use foo::*`	Re-exports all items	Flattening APIs
`pub use foo::Item`	Re-exports specific item	Selective exposure

Crate Responsibilities

Module	Crate	Purpose
`grep::cli`	`grep_cli`	Terminal utilities
`grep::matcher`	`grep_matcher`	Core `Matcher` trait
`grep::regex`	`grep_regex`	Rust regex implementation
`grep::pcre2`	`grep_pcre2`	PCRE2 implementation (optional)
`grep::searcher`	`grep_searcher`	File searching coordination
`grep::printer`	`grep_printer`	Output formatting

Feature Flag Pattern

```
// In Cargo.toml:
[features]
default = []
pcre2 = ["grep_pcre2"]

// In lib.rs:
#[cfg(feature = "pcre2")]
pub extern crate grep_pcre2 as pcre2;
```