Higher Concept: Default Create

Creation of objects is a concept higher than any language implementation of the concept.

Universal Default Creation

- Default creation is REQUIRED
- Non-default creation overrides default
- Non-default creation must be specified
- Otherwise—default is used
- Default creation is code—NOT compiler!
  - Why?

Example: Java no-arg constructor

- “compiler automatically provides …”
  - The “code” in the compiler is unreachable by you!
  - You (programmer) CANNOT override this code!
- Poor design
  - Removes your capacity to create universal default

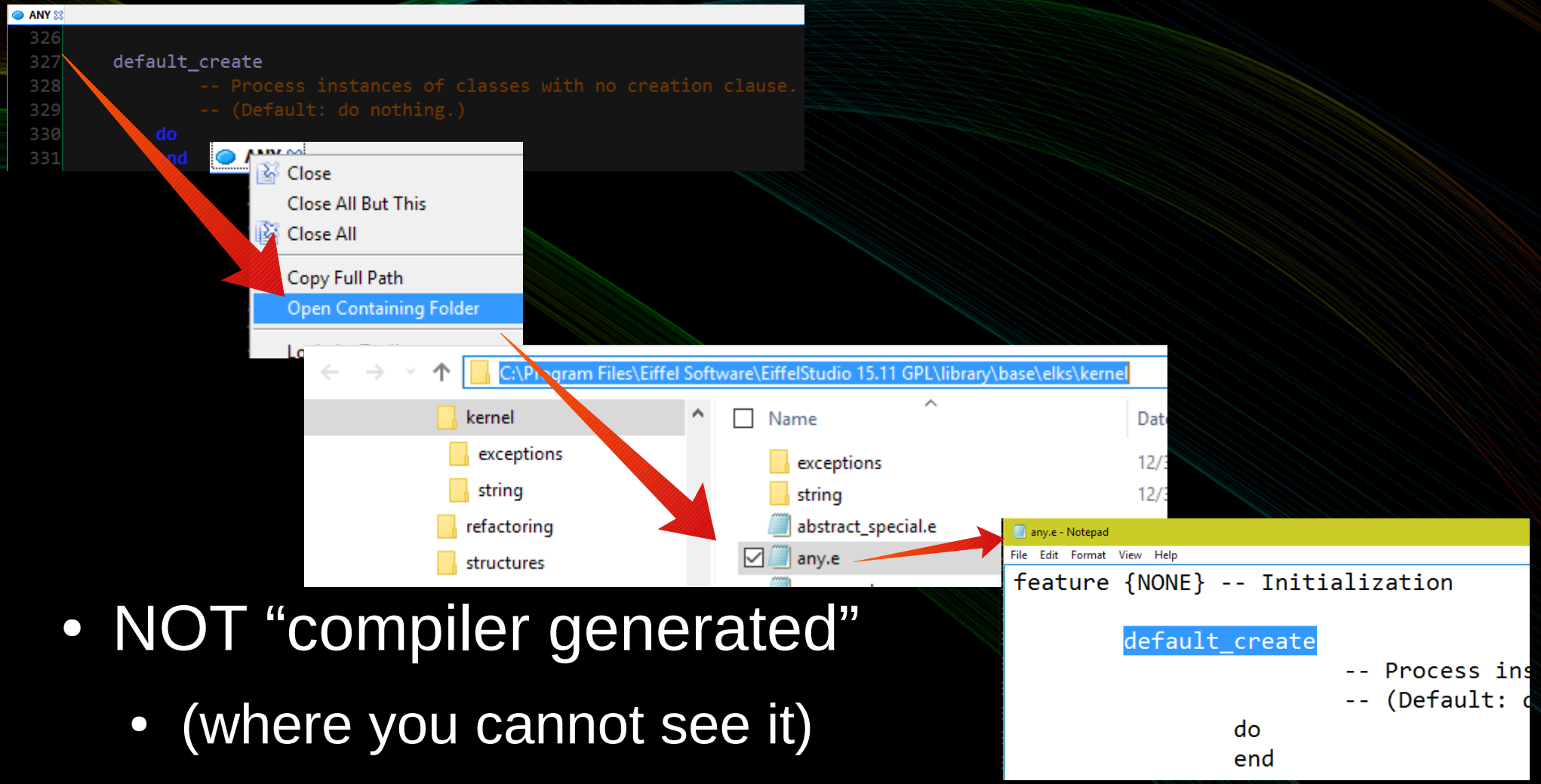
`Bicycle yourBike = new Bicycle();` invokes the no-argument constructor to create a new `Bicycle` object called `yourBike`.

- Complains = BUG

You don't have to provide any constructors for your class, but you must be careful when doing this. The compiler automatically provides a no-argument, default constructor for any class without constructors. This default constructor will call the no-argument constructor of the superclass. In this situation, the compiler will complain if the superclass doesn't have a no-argument constructor so you must verify that it does. If your class has no explicit superclass, then it has an implicit superclass of `Object`, which *does* have a no-argument constructor.

Example: Eiffel Default Create

- ANY = file → You can “override” as you need!



- NOT “compiler generated”
  - (where you cannot see it)

Higher Concept: Default Create

Creation of objects is a concept higher than any language implementation of the concept.

Universal Default Creation

- Default creation is REQUIRED
- Non-default creation overrides default
- Non-default creation must be specified
- Otherwise—default is used
- Default creation is code—NOT compiler!
  - Why?

Example: Java no-arg constructor

- “compiler automatically provides ...”
  - The “code” in the compiler is unreachable by you!
  - You (programmer) CANNOT override this code!
- Poor design
  - Removes your capacity to create universal default

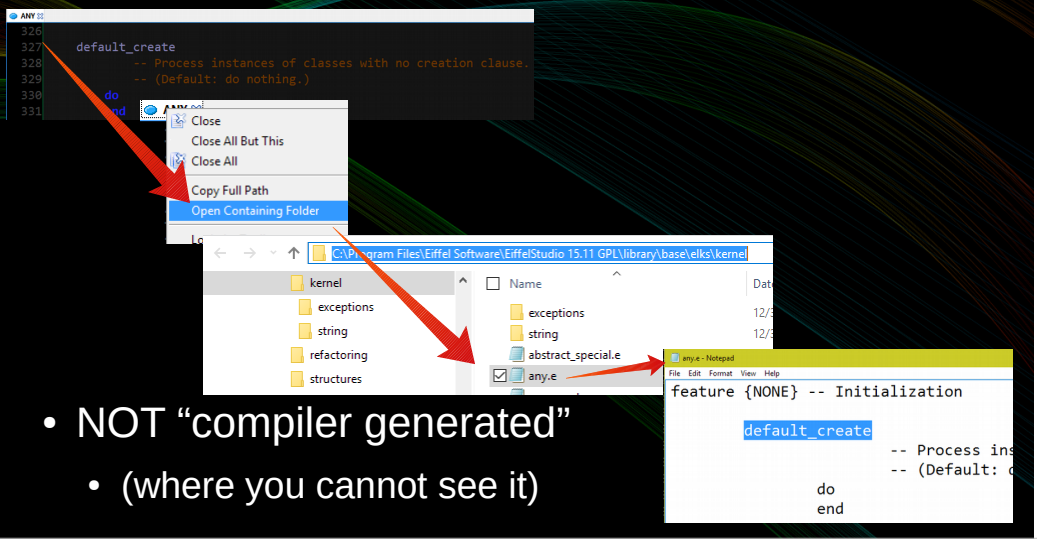
`Bicycle yourBike = new Bicycle();` invokes the no-argument constructor to create a new `Bicycle` object called `yourBike`.

- Complains = BUG

You don't have to provide any constructors for your class, but you must be careful when doing this. The compiler automatically provides a no-argument, default constructor for any class without constructors. This default constructor will call the no-argument constructor of the superclass. In this situation, the compiler will complain if the superclass doesn't have a no-argument constructor so you must verify that it does. If your class has no explicit superclass, then it has an implicit superclass of `Object`, which ***does*** have a no-argument constructor.

Example: Eiffel Default Create

- ANY = file → You can “override” as you need!



- NOT “compiler generated”
  - (where you cannot see it)