

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!

The screenshot illustrates the location of the `ANY` file in the EiffelStudio library structure. It shows a code editor window with the `default_create` routine, a context menu with `Open Containing Folder` selected, a file explorer window showing the path `C:\Program Files\Eiffel Software\EiffelStudio 15.11 GPL\library\base\elks\kernel`, and a Notepad window showing the `any.e` file content.

```
326
327 default_create
328 -- Process instances of classes with no creation clause.
329 -- (Default: do nothing.)
330 do
331 end
```

File Explorer Path: `C:\Program Files\Eiffel Software\EiffelStudio 15.11 GPL\library\base\elks\kernel`

File Explorer Contents:

- kernel
- exceptions
- string
- refactoring
- structures
- exceptions
- string
- abstract_special.e
- ☒ any.e

any.e - Notepad

```
feature {NONE} -- Initialization

  default_create

  -- Process instances of classes with no creation clause.
  -- (Default: do nothing.)

do
end
```

- 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!

The screenshot illustrates the process of locating and overriding the `default_create` routine in the `ANY` file. It shows three overlapping windows:

- ANY.i**: A code editor window showing the `default_create` routine definition. The code is as follows:

```
326  
327 default_create  
328 -- Process instances of classes with no creation clause.  
329 -- (Default: do nothing.)  
330 do  
331 end
```
- File Explorer**: A window showing the file system path `C:\Program Files\Eiffel Software\EiffelStudio 15.11 GPL\library\base\elks\kernel`. The `kernel` folder is expanded, showing subfolders like `exceptions`, `string`, `refactoring`, and `structures`, along with files like `abstract_special.e` and `any.e`. The `any.e` file is selected.
- any.e - Notepad**: A Notepad window showing the `any.e` file. The `default_create` routine is highlighted in blue. The code is as follows:

```
feature {NONE} -- Initialization  
  
  default_create  
    -- Process instances of classes with no creation clause.  
    -- (Default: do nothing.)  
  do  
  end
```

Red arrows indicate the flow of information: from the `ANY.i` window to the File Explorer, and from the File Explorer to the `any.e` Notepad window.

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