# Final Thoughts..


- Final Project Deliverables
  - Documentation
  - Video demo (at most 5 minutes)
    - Due on or before June 20
    - Send github link containing the files to edujzenitram@gmail.com
- Final Exam
  - Coding

# Design Patterns

“If a problem occurs over and over again, a solution to that problem has been used effectively.
That solution is described as a pattern.”

# What is the Design Pattern?

- A description of a problem and its solution that you can apply to many programming situations
- Language-independent strategies for solving common object-oriented design problems.

# Do I have to use Design Patterns?

- If you want to be a professional Java developer, you should know at least some popular solutions to coding problems.
- Learning design patterns speeds up your experience accumulation in OOA/OOD.
- You will be able to use these to communicate with your fellows or assessors more effectively.

# How many Design Patterns?

- Many.
- At least 250 existing patterns are used in OO world
  - Spaghetti which refers to poor coding habits
  - The 23 design patterns by GOF are well known

- Note that the design patterns are not idioms or algorithms or components.

# What is the relationship among these patterns?

- Generally, to build a system, you may need many patterns to fit together. Different designers may use different patterns to solve the same problem.
- Usually:
  - Some patterns naturally fit together
  - One pattern may lead to another
  - Some patterns are similar and alternative
  - Patterns are discoverable and documentable
  - Patterns are not methods or framework
  - Patterns give you hint to solve a problem effectively

# The Pattern Concept

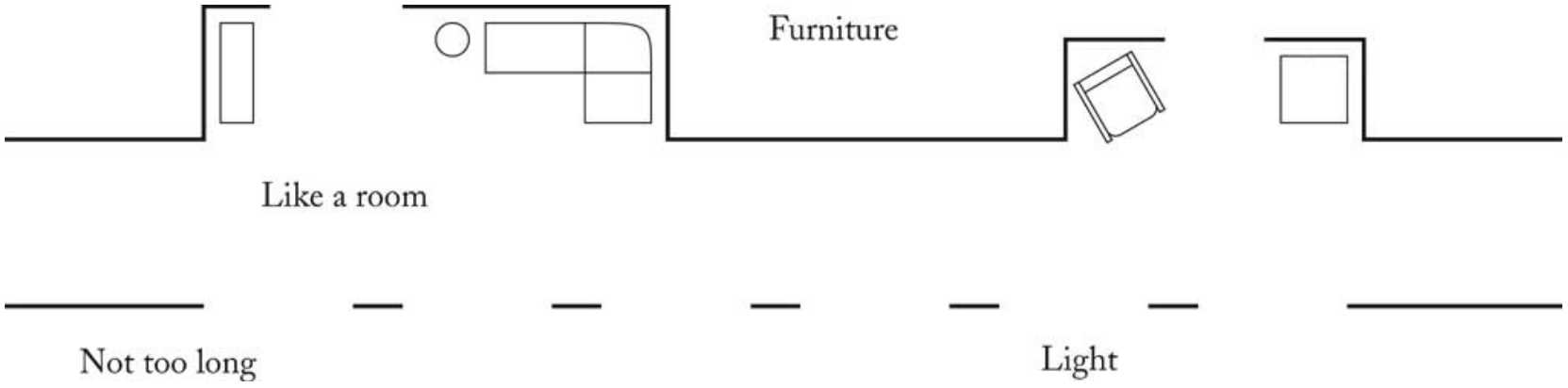
- Each pattern has
  - a short *name*
  - a brief description of the *context*
  - a lengthy description of the *problem*
  - a prescription for the *solution*

# Short Passages Pattern

- **Context**
  - "...Long, sterile corridors set the scene for everything bad about modern architecture...“
- **Problem**
  - A description of the problem, including
    - a depressing picture
    - issues of light and furniture
    - research about patient anxiety in hospitals
    - research that suggests that corridors over 50 ft are considered uncomfortable

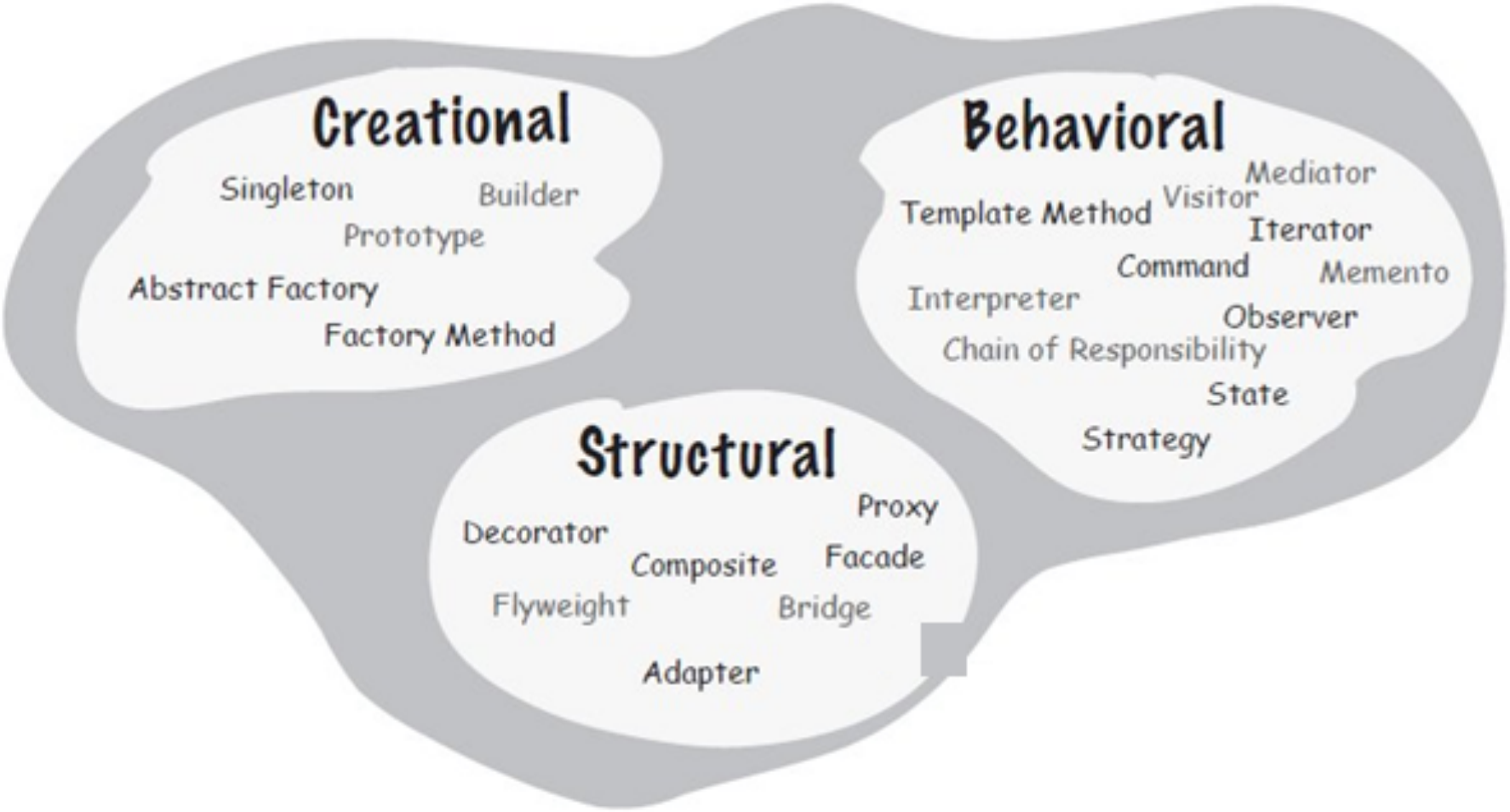
# Short Passages Pattern

- **Solution**
  - Keep passages short. Make them as much like rooms as possible, with carpets or wood on the floor, furniture, bookshelves, beautiful windows. Make them generous in shape and always give them plenty of light; the best corridors and passages of all are those which have windows along an entire wall.



# Types of Patterns

- Creational
  - Object creation
- Structural
  - Relationship between entities
- Behavioral
  - Communication between Objects




http://www.blackwasp.co.uk/gofpatterns.aspx

# Creational Design Patterns (GoF)

- Creational patterns provide ways to instantiate single objects or groups of related objects.
  - Abstract Factory
  - Builder
  - Factory
  - Prototype
  - Singleton

# Factory Pattern

# Factory Pattern

- In the real world, a factory is a place where products are manufactured.
- In software, Factory Method is a GoF creational design pattern used for creating objects
- The design pattern is used widely in Java API

# Factory Pattern

- A Factory Pattern or Factory Method Pattern says that just **define an interface or abstract class for creating an object but let the subclasses decide which class to instantiate.**
  - In other words, subclasses are responsible to create the instance of the class.
- The Factory Method Pattern is also known as **Virtual Constructor.**

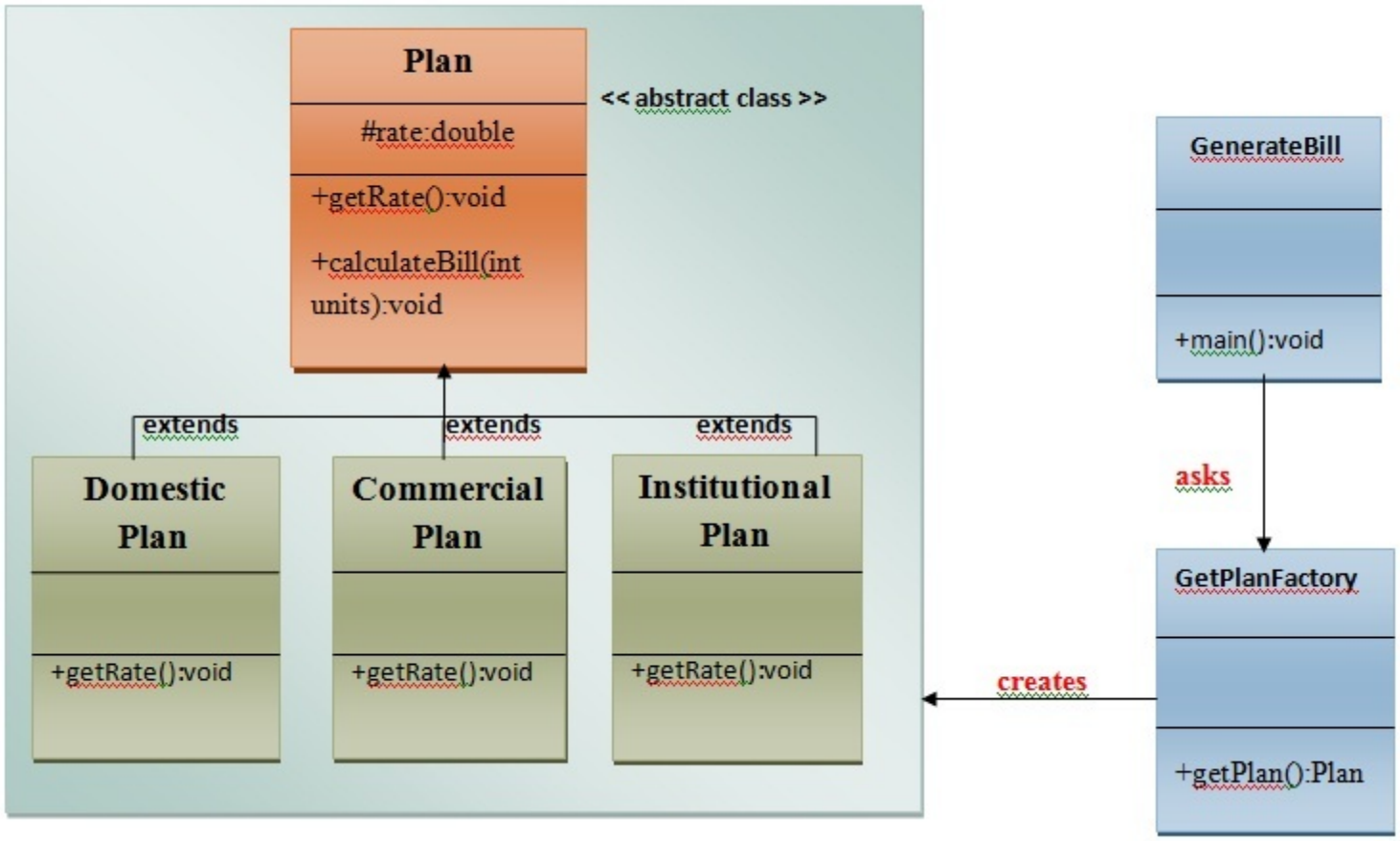
# Advantage

- Factory Method Pattern allows the sub-classes to choose the type of objects to create.
- It promotes **loose-coupling** by eliminating the need to bind application-specific classes into the code.
  - That means the code interacts solely with the resultant interface or abstract class, so that it will work with any classes that implement that interface or that extends that abstract class.

# Usage

- When a class doesn't know what sub-classes will be required to create
- When a class wants that its sub-classes specify the objects to be created.
- When the parent classes choose the creation of objects to its sub-classes

# UML for Factory Method Pattern (Example)

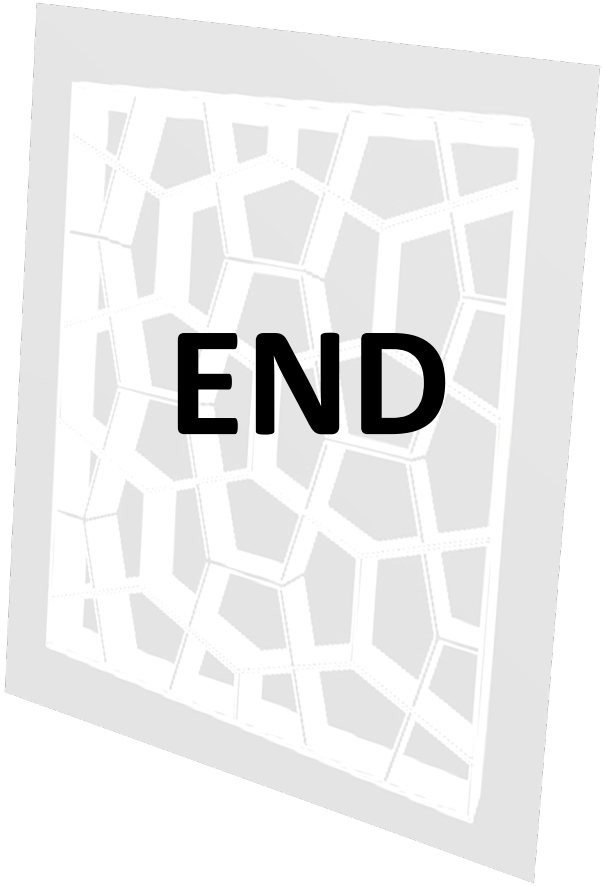


## Calculate Electricity Bill: A Real World Example of Factory Method

- **Step 1:** Create a Plan abstract class.
- **Step 2:** Create the concrete classes that extends Plan abstract class.
- **Step 3:** Create a GetPlanFactory to generate object of concrete classes based on given information
- **Step 4:** Generate Bill by using the GetPlanFactory to get the object of concrete classes by passing an information such as type of plan DOMESTICPLAN or COMMERCIALPLAN or INSTITUTIONALPLAN.

# To do

1. Group yourselves by 3
2. Update the xls sheet provided in the WAG
3. Prepare a tutorial video (@most 10 minutes )
4. Send github link containing the files to edujzenitram@gmail.com by June 16, 2020



# END