МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

КАФЕДРА «ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ»

Лабораторная работа № 4

По предмету «Объектно-Ориентированное Программирование и Проектирование»

Тема: «Введение в Паттерны»

Выполнил:

студент 3-го курса

группы АС-59  
 Качан Д. С.

Проверил:

Михняев.Е.В

Брест 2022

**Цель:** Получить навыки применения паттернов абстрактная фабрика и прототип.

**from abc import ABC, abstractmethod**

**import pygame**

**class Prototype(ABC):**

**@abstractmethod**

**def clone(self):**

**pass**

**class Rectangle(ABC):**

**@abstractmethod**

**def draw(self, surface: pygame.Surface):**

**pass**

**class SimpleRectangle(Rectangle, Prototype):**

**def \_\_init\_\_(self, x: int, y: int, width: int, height: int):**

**super().\_\_init\_\_()**

**self.x = x**

**self.y = y**

**self.width = width**

**self.height = height**

**def draw(self, surface: pygame.Surface):**

**pygame.draw.rect(surface, (0, 0, 0), (self.x, self.y, self.width, self.height))**

**def clone(self) -> 'SimpleRectangle':**

**return SimpleRectangle(self.x, self.y, self.width, self.height)**

**class RoundedRectangle(Rectangle, Prototype):**

**def \_\_init\_\_(self, x: int, y: int, width: int, height: int):**

**super().\_\_init\_\_()**

**self.x = x**

**self.y = y**

**self.width = width**

**self.height = height**

**def draw(self, surface: pygame.Surface):**

**pygame.draw.rect(surface, (0, 0, 0), (self.x, self.y, self.width, self.height), border\_radius=10)**

**def clone(self) -> 'RoundedRectangle':**

**return RoundedRectangle(self.x, self.y, self.width, self.height)**

**class Ellipse(ABC):**

**@abstractmethod**

**def draw(self, surface: pygame.Surface):**

**pass**

**class SimpleEllipse(Ellipse, Prototype):**

**def \_\_init\_\_(self, x: int, y: int, width: int, height: int):**

**super().\_\_init\_\_()**

**self.x = x**

**self.y = y**

**self.width = width**

**self.height = height**

**def draw(self, surface: pygame.Surface):**

**pygame.draw.ellipse(surface, (0, 0, 0), (self.x, self.y, self.width, self.height))**

**def clone(self) -> 'SimpleEllipse':**

**return SimpleEllipse(self.x, self.y, self.width, self.height)**

**class Line(ABC):**

**@abstractmethod**

**def draw(self, surface: pygame.Surface):**

**pass**

**class SimpleLine(Line, Prototype):**

**def \_\_init\_\_(self, x1: int, y1: int, x2: int, y2: int):**

**super().\_\_init\_\_()**

**self.x1 = x1**

**self.y1 = y1**

**self.x2 = x2**

**self.y2 = y2**

**def draw(self, surface: pygame.Surface):**

**pygame.draw.line(surface, (0, 0, 0), (self.x1, self.y1), (self.x2, self.y2), 5)**

**def clone(self) -> 'SimpleLine':**

**return SimpleLine(self.x1, self.y1, self.x2, self.y2)**

**class Text(ABC):**

**@abstractmethod**

**def draw(self, surface: pygame.Surface):**

**pass**

**class SimpleText(Text, Prototype):**

**def \_\_init\_\_(self, text: str, center\_x: int, center\_y: int):**

**self.text = text**

**self.x = center\_x**

**self.y = center\_y**

**self.font = pygame.font.Font('Roboto-Regular.ttf', 14)**

**def draw(self, surface: pygame.Surface):**

**text\_object = self.font.render(self.text, True, (255, 255, 255), (0, 0, 0))**

**text\_rect = text\_object.get\_rect()**

**text\_rect.center = (self.x, self.y)**

**surface.blit(text\_object, text\_rect)**

**def clone(self) -> 'SimpleText':**

**return SimpleText(self.text, self.x, self.y)**

**class DiagramFactory:**

**def \_\_init\_\_(self,**

**rectangle: Rectangle | Prototype,**

**ellipse: Ellipse | Prototype,**

**line: Line | Prototype,**

**text: Text | Prototype):**

**super().\_\_init\_\_()**

**self.rectangle\_prototype = rectangle**

**self.ellipse\_prototype = ellipse**

**self.line\_prototype = line**

**self.text\_prototype = text**

**def create\_rectangle(self) -> Rectangle:**

**return self.rectangle\_prototype.clone()**

**def create\_ellipse(self) -> Ellipse:**

**return self.ellipse\_prototype.clone()**

**def create\_line(self) -> Line:**

**return self.line\_prototype.clone()**

**def create\_text(self) -> Text:**

**return self.text\_prototype.clone()**

**if \_\_name\_\_ == '\_\_main\_\_':**

**pygame.init()**

**\_rectangle: Rectangle = RoundedRectangle(100, 50, 500, 200)**

**\_ellipse: Ellipse = SimpleEllipse(300, 500, 500, 200)**

**\_line: Line = SimpleLine(350, 250, 550, 500)**

**\_text: Text = SimpleText("Sample text", 350, 150)**

**factory: DiagramFactory = DiagramFactory(\_rectangle, \_ellipse, \_line, \_text)**

**rectangle: Rectangle = factory.create\_rectangle()**

**ellipse: Ellipse = factory.create\_ellipse()**

**line: Line = factory.create\_line()**

**text: Text = factory.create\_text()**

**objects = [rectangle, ellipse, line, text]**

**window = pygame.display.set\_mode((1920, 1080))**

**clock = pygame.time.Clock()**

**is\_working = True**

**while is\_working:**

**for event in pygame.event.get():**

**if event.type == pygame.KEYDOWN:**

**if event.key == pygame.K\_ESCAPE:**

**is\_working = False**

**window.fill((255, 255, 255))**

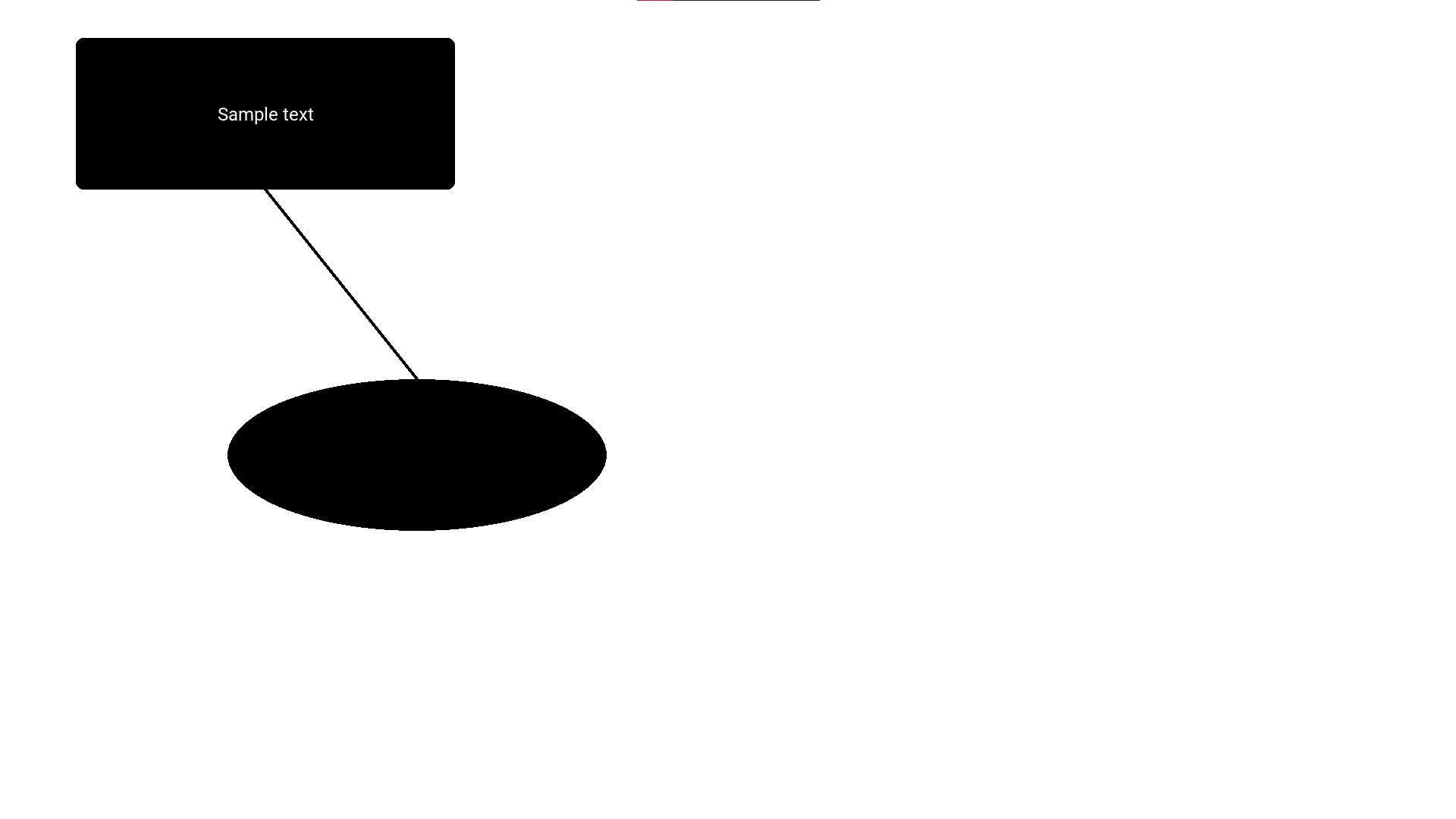
**for obj in objects:**

**obj.draw(window)**

**pygame.display.flip()**

**clock.tick(60)**

Результат работы программы:



**Вывод:** Получил навыки применения паттернов абстрактная фабрика и прототип.