**Homework 5**

**COM S 362**

**Fall 2021**

In the following problems you are asked to create standard UML diagrams that describe the code provided at the end of this document. The code is written in a fictional language. The syntax is different than Java, but assume classes, interfaces, methods, implements, extends, public, private etc. have the same meaning and behavior as in Java.

No hand drawn diagrams will be accepted, use of a software drawing tool is required (Umlet is recommended but not required). Some UML tools automatically convert code into diagrams, the diagrams they produce can be messy and poor at conveying the intention of a design, for this assignment they are not allowed.

1. **(30 points)** Create a communication diagram that describes the behavior of the program starting with the call to startApp() in AppBoostrap as the initiating message. Assume the platform is “PC”. Include only relevant objects in the diagram. Do not include objects of common libraries (e.g., LinkedList). Use the flat numbering scheme described by Folwer in the reading.

Diagram

Description automatically generated

1. **(30 points)** Create a sequence diagram that describes the behavior of the program starting with the call to startApp() in AppBootstrap as the founding message. Assume the platform is “PC”. Include only the relevant objects in the diagram. Do not include objects of common libraries (e.g., LinkedList). For this diagram, do not use interaction frames to describe decisions or loops, instead show specific messages to specific objects that result from their execution.

Diagram

Description automatically generated with medium confidence

1. **(40 points)** Create a class diagram that includes all classes and interfaces, except for those of common libraries (e.g., LinkedList). The diagram should be “fully-dressed”, meaning at a minimum, it must include wherever appropriate:

A picture containing text, sky

Description automatically generated

* + visibility indicators
  + associations
  + dependencies
  + generalizations
  + implementations
  + multiplicities
  + attributes (properties)
  + operations (methods)
  + parameters
  + return types

**interface** WidgetFactory public **method** Button createButton() public **method** Checkbox createCheckbox()

**interface** Widget public **method** paint()

**class** Button implements Widget public **method** setText(String text) public **method** String getText()

**class** Checkbox implements Widget public **method** setSelected(Boolean is) public **method** Boolean getSeletect()

**class** PCButton extends Button public **method** paint()

// render button PC style

**class** MobileButton extends Button public method paint()

// render button mobile style

**class** PCCheckbox extends Checkbox public **method** paint()

// render checkbox PC style

**class** MobileCheckbox extends Checkbox public **method** paint()

// render checkbox mobile style

**class** PCFactory implements WidgetFactory public **method** Button createButton() return new PCButton() public **method** Checkbox createCheckbox() return new PCCheckbox()

**class** MobileFactory implements WidgetFactory public **method** Button createButton() return new MobileButton() public **method** Checkbox createCheckbox() return new MobileCheckbox()

**class** App

private WidgetFactory factory private LinkedList<Widget> widgets public **constructor** App(WidgetFactory factory) this.factory = factory widgets = new LinkedList<Widget>() public **method** createUI()

Checkbox cb = factory.createCheckbox()

cb.setSelected(false)

Button b = factory.createButton()

b.setText("Click Here”) widgets.add(cb) widgets.add(b) public **method** paint()

// render each widget **for** (Widget w : widgets)

w.paint() **class** AppBootstrap

public **constructor** AppConfig()

// init an AppConfig

public **method** startApp()

String platform = getPlatform()

Factory factory **if** (platform == "PC") factory = new PCFactory() **else if** (platform == "Mobile") factory = new MobileFactory()

App app = new App(factory) app.createUI() app.paint()

private **method** String getPlatform()

// returns platform

public static **method** main()

AppBootstrap boot = new AppBootstrap()

Boot.startApp()