**Вопросы к лабораторной работе №4-5**

1. Перечислите преимущества и недостатки WPF?

Из явных достоинств WPF можно выделить следующее:

1. WPF поддерживает гибкий поток, размещающий элементы управления на основе их содержимого вместо того, чтобы фиксировать элементы управления на одном постоянном месте с какими-то постоянными определёнными координатами.

2. Поддерживает аудио и видео.

3. Платформа не нуждается ни в каких таймерах что бы форма перерисовала себя. Вместо этого всего WPF поддерживает анимацию, которая запускается автоматически.

4.Работая в WPF, вы имеете дело с базовыми фигурами, блоками текста и прочими контроллерами, что полностью упрощает задачу в рисовании интерфейса: нет необходимости рисовать пиксели вручную.

5. Разрешено редактировать интерфейс напрямую из кода XAML.

6. Использование стилей и шаблонов

Недостатки:

1. Более слабая поддержка в дизайнере окон. Много приходится вводить руками в XAML-код.

2. Сглаженный шрифт.

3. Отсутствуют некоторые полезные графические возможности, которые были в GDI/GDI+. В частности, в WPF нет операций ROP2, не поддерживаются наконечники линий и нет составных линий (например нельзя нарисовать двойную линию).

4. Трудно нарисовать собственный элемент управления со стандартным оформлением какого-либо системного контрола (в WinForms для этого были специальные классы с методами рендеринга).

5. Нет автоматического затенения рисунков на отключенных (disabled) кнопках, приходится это делать вручную.

6. Неудачные названия многих классов, из-за чего полная каша и путаница с названиями.

7. Плохая документированность. Многие тривиальные вещи приходится узнавать, ища ответ в интернете.

8. Отсутствуют такие элементы управления, как NumericUpDown и PropertyGrid.

9. Отсутствует HelpProvider.

10. Чувствительность к драйверам видеокарты (на старых версиях может глючить).

11. Нормально работает только под Windows XP/Vista/7.

12. Ну и наконец, идеология WPF в корне отличается от всего, что было раньше. Придется учиться всему заново. Причем, "немножко подучиться" не получится. Нужно изучать всё - с самого начала и до конца.

2. Зачем нужен язык XAML? Каким образом он используется в WPF?

XAML — это язык разметки, который появился вместе с первой версией WPF от Microsoft.

3. Какие бывают контейнеры компоновки?

В WPF **компоновка** осуществляется при помощи специальных **контейнеров**. Фреймворк предоставляет нам следующие **контейнеры**: Grid, UniformGrid, StackPanel, WrapPanel, DockPanel и Canvas.

4. Перечислите основные группы элементов управления. Расскажите про объектную модель WPF.

Все элементы управления могут быть условно разделены на несколько подгрупп:

* **Элементы управления содержимым**, например кнопки (Button), метки (Label)
* **Специальные контейнеры**, которые содержат другие элементы, но в отличие от элементов Grid или Canvas не являются контейнерами компоновки - ScrollViewer,GroupBox
* **Декораторы**, чье предназначение создание определенного фона вокруг вложенных элементов, например, Border или Viewbox.
* **Элементы управления списками**, например, ListBox, ComboBox.
* **Текстовые элементы управления**, например, TextBox, RichTextBox.
* **Элементы, основанные на диапазонах значений**, например, ProgressBar, Slider.
* **Элементы для работ с датами**, например, DatePicker и Calendar.
* **Остальные элементы управления**, которые не вошли в предыдущие подгруппы, например, Image.

**Windows** **Presentation** **Foundation** (**WPF**) представляет собой обширный API-интерфейс для создания настольных графических программ, имеющих насыщенный дизайн и интерактивность. В отличие от устаревшей технологии Windows Forms, **WPF** включает новую **модель** построения пользовательских приложений.

5. Объясните назначение класса DependencyProperty. Зачем нужны свойства зависимостей? Как создать новой свойство зависимости и в каких случаях это необходимо?

**Dependency** **Property** (Свойство зависимости) — это основа любого Silverlight/WPF/WP7 приложения. Их можно рассматривать как обычные свойства, но они обладают дополнительными возможностями, позволяющими работать с анимацией, привязкой данных, стилями. Также они эффективно потребляют память и поддерживают уведомления об изменениях.

Задачей свойств зависимостей является предоставление систематического способа расчета значения свойства на основании других входных данных (других свойств, событий и состояний, которые возникают в вашем приложении во время выполнения). Входными данными могут быть:

* Внешние входные данные, например настройка пользователя
* Механизмы оперативного определения свойств, например привязка данных, анимация и раскадровка
* Шаблоны многократного использования, например ресурсы и стили
* Значения, имеющие родительско-дочернюю связь с другими элементами дерева объектов

Свойство зависимостей представляет или поддерживает особую функцию модели программирования для определения приложений среды выполнения Windows, которые используют XAML для описания пользовательского интерфейса и расширения компонентов C#, Microsoft Visual Basic или Visual C++ (C++/CX) для создания программного кода. Эти функции включают перечисленные ниже.

* привязка данных,
* стили.
* Раскадрованные анимации
* Поведение «PropertyChanged»; свойство зависимостей может быть реализовано для предоставления обратных вызовов, распространяющих изменения на другие свойства зависимостей
* Использование значения по умолчанию, которое поступает из метаданных свойства
* Общее системное средство для работы со свойствами, например [**ClearValue**](https://docs.microsoft.com/ru-ru/uwp/api/windows.ui.xaml.dependencyobject.clearvalue), и поиск по метаданным

6. Каким образом осуществляется обработка событий в WPF? Что означает концепция маршрутизированных событий? Перечислите основные группы событий.

Маршрутизируемые события — это события с большими транспортными возможностями: они могут туннелироваться вниз и распространяться пузырьками наверху по дереву элементов и по пути запускать обработчики событий.

Модель событий WPF отличается от событий WinForms не только декларативным подключением. События, возникнув на одном элементе, могут обрабатываться на другом. События могут подниматься и опускаться по дереву элементов.

Так, маршрутизируемые события делятся на три вида:

* **Прямые** (direct events) - они возникают и отрабытывают на одном элементе и никуда дальше не передаются. Действуют как обычные события.
* **Поднимающиеся** (bubbling events) - возникают на одном элементе, а потом передаются дальше к родителю - элементу-контейнеру и далее, пока не достигнет наивысшего родителя в дереве элементов.
* **Опускающиеся, туннельные** (tunneling events) - начинает отрабатывать в корневом элементе окна приложения и идет далее по вложенным элементам, пока не достигнет элемента, вызвавшего это событие.

Для элементов управления в WPF определено большое количество событий, которые условно можно разделить на несколько групп:

* События клавиатуры
* События мыши
* События стилуса
* События сенсорного экрана/мультитач
* События жизненного цикла

7. Что такое Resource Dictionary?

Словарь ресурсов представляет собой просто XAML-документ, который всего лишь хранит необходимые ресурсы.

8. Что такое стиль и как его создать? В чем преимущество использования стиля?

**Стиль** — это именованный и сохраненный набор параметров форматирования. Форматировать текст с помощью **стиля** намного быстрее, чем изменять вручную каждый элемент форматирования, к тому же при этом гарантируется единообразие внешнего.

9. Что такое Command? Расскажите паттерн Command. Как в WPF используется Command и для чего?

**Command** — это команда, с которой связан объект CommandBinding.

Модель команд в WPF состоит из четырех аспектов:

* Сама **команда**, которая представляем выполняемую задачу
* **Привязка команд**, которая связывает команду с определенной логикой приложения
* **Источник команды** - элемент пользовательского интерфейса, который запускает команду (например, кнопка, по нажатию который выполняется команда)
* **Цель команды** - элемент интерфейса, на котором выполняется команда

**Команда** — это поведенческий паттерн проектирования, который превращает запросы в объекты, позволяя передавать их как аргументы при вызове методов, ставить запросы в очередь, логировать их, а также поддерживать отмену операций.