1. Компоненты пользовательского интерфейса. Иерархия компонентов.

WIMP

● Window

● Icon

● Menu

● Pointer

11 import

2.Базовые классы  
2.1.Component  
Класс java.awt.Component  
☑ Компонент (widget, control) — отображаемый и  
взаимодействующий с пользователем элемент GUI  
● java.awt.Component - абстрактный класс — элемент GUI  
● цвет, размер, местоположение  
● порождает основные события

☑ Класс Color  
-----  
 Положение и размер  
☑ Класс Point (int x, int y)  
☑ Класс Dimension (int height, int width)  
☑ Класс Rectangle (int x, int y, int height, int width)  
-----  
 Шрифт  
☑ Класс Font  
  
Методы класса Component  
 Цвет текста и цвет фона  
Color getForeground()  
void setForeground(Color)  
Color getBackground()  
void setBackground(Color)  
 Положение и размеры  
void setBounds(Rectangle)  
Rectangle getBounds()  
void setLocation(Point)  
Point getLocation()  
void setSize(Dimension)  
Dimension getSize()  
 Шрифт  
Font getFont()  
void setFont(Font)  
 Видимость  
boolean isVisible()  
void setVisible(boolean)  
 Активность  
boolean isEnabled()  
void setEnabled(boolean)  
 Дополнительное рисование  
void paint(Graphics)  
void update(Graphics)  
void repaint()  
Graphics — графический контекст компонента  
☑ Системный вызов paint  
● первое отображение  
● изменение размера  
● необходимость перерисовки  
☑ JVM вызывает paint(Graphics)  
☑ Программный вызов paint  
● изменение состояния компонента  
☑ в программе вызывается repaint()  
☑ регистрируется событие отрисовки  
☑ JVM вызывает update(Graphics)

2.2 Container

Класс java.awt.Container

☑ Контейнер — компонент, который содержит другие компоненты ● extends java.awt.Component

☑ Иерархия компонентов – дерево

☑ Методы: ● add(Component) ● setLayout(LayoutManager) ● validate()

Размещение компонентов в контейнере

☑ Абсолютное позиционирование

● Отсутствует реакция на изменение размера контейнера

● Проблемы с изменением шрифта или локали

☑ Менеджер компоновки

● Управляет расположением и размером компонентов

Менеджер компоновки

☑ Интерфейс LayoutManager

● Container.setLayout(LayoutManager)

● Container.add(Component)

☑ Интерфейс LayoutManager2

● Container.setLayout(LayoutManager2, Object constraints) ● Container.add(Component, Object constraint

☑ Расстановка элементов

● Container.validate()

● Container.invalidate()

● Container.doLayout()

● LayoutManager.layoutContainer(Container)

☑ Управление размером компонентов

● Component.getPreferredSize()

● Component.getMinimumSize()

● Component.getMaximumSize()

2.3JComponent

extends java.awt.Container — может содержать картинку ☑ всплывающие подсказки — setToolTipText() ☑ построение основано на шаблоне MVC ☑ встроенная двойная буферизация при отрисовке ☑ реализация метода paint ● Для отрисовки нужно переопределить paintComponent(g) ● Необходимо вызывать super.paintComponent(g);

3. Менеджеры компоновки

☑ BoxLayout ● Компоненты располагаются в один ряд вертикально или горизонтально ☑ Класс Box — контейнер с BoxLayout ● Box.createHorizontalBox() ● Box.createVerticalBox() ☑ createRigidArea(Dimension) ☑ createHorizontalGlue() ☑ createVerticalGlue() ☑ Filler(minSize, prefSize, maxSize) – заполнитель

☑ GroupLayout ● Все компоненты описываются дважды — горизонтальное расположение и вертикальное расположение ● Все компоненты являются участниками групп — последовательных и параллельных

☑ SpringLayout ● Все компоненты соединены пружинами (Spring), которые имеют минимальную, максимальную и предпочтительную длину ● Обычно используется автоматическими расстановщиками

JLabel

☑ Метка изначально прозрачная ☑ метод setOpaque(true) — сделать непрозрачной

4. Модель обработки событий. Класс-слушатель и класс-событие.

Обработка событий

☑ Событийно-ориентированное программирование

☑ Не задана последовательность выполнения кода

☑ Код выполняется асинхронно при наступлении определенных событий

☑ Шаблон Observer

☑ button.addActionListener(label) ☑ Перевод: – Кнопка, когда тебя нажмут — скажи метке – Ладно, записала...

☑ Источник события — любой компонент ☑ Событие — потомок класса AWTEvent ☑ Обработчик реализует интерфейс ...Listener и его методы ☑ Методу передается объект события для обработки

class A implements ActionListener { Button b = new Button("OK"); Label l = new Label("Button pressed"); l.setVisible(false); b.addActionListener(this); - подписка на событие ...... public void actionPerformed(ActionEvent e) { l.setVisible("true"); - реакция на событие }

☑ Анонимным классом b.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent e) { l.setVisible("true"); } }); ☑ Лямбда-выражением b.addActionListener((e) -> l.setVisible("true"));

5Технология JavaFX. Особенности архитектуры, отличия от AWT / Swing.

<https://pianalytix.com/difference-b-w-awt-and-swing-and-javafx/>

6.

Internationalization describes **designing a product in a way that it may be readily consumed across multiple countries**. This process is used by companies looking to expand their global footprint beyond their own domestic market understanding consumers abroad may have different tastes or habits.