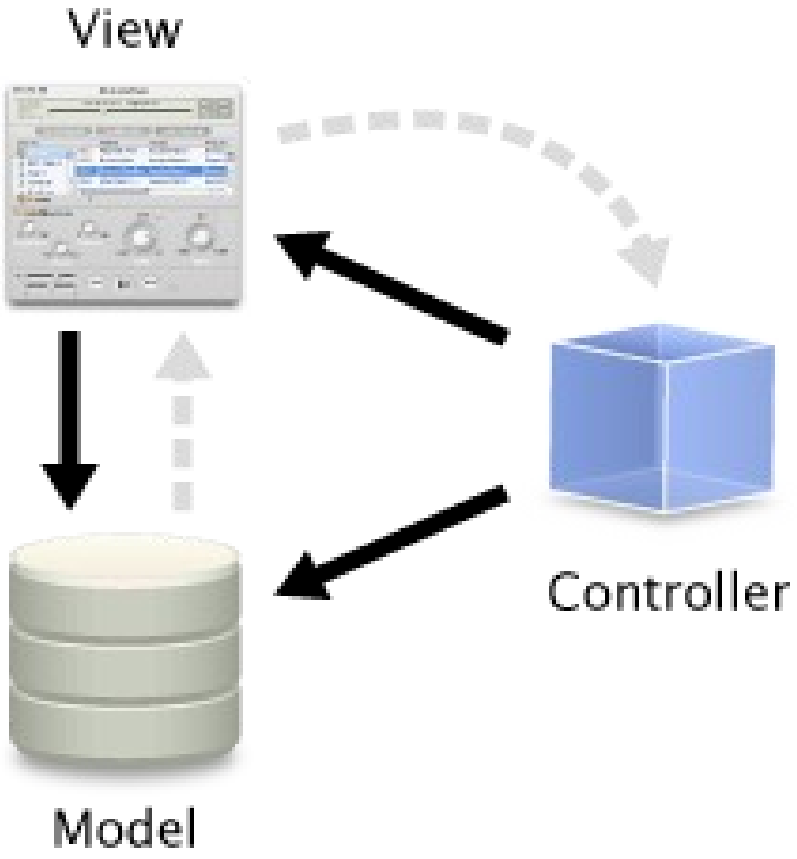
MVC

MVC

- Arquitetura de Software
- Isola:
  - Domínio Lógico
    - Lógica da aplicação para o usuário
  - Interface do usuário
    - Input e apresentação
- Permite:
  - Desenvolvimento, teste e manutenção independentes

MVC

- MVC – Model View Controller



MVC

- Model
  - Gerencia o comportamento e dados no domínio da aplicação
  - Responde a requisições de informações sobre seu estado
    - Normalmente da View
  - Responde a instruções de mudança de estado
    - Normalmente do Controller
  - Em sistemas orientados a eventos, o Model notifica os observadores (View) quando há alteração de informações

MVC

- View
  - Renderiza o modelo
    - Normalmente em um elemento de Interface de usuário
  - Múltiplas Views podem existir para um único Model
    - Normalmente a interface tem uma correspondência de 1 pra 1
  - Em Java as Views são feitas com componentes AWT ou Swing

MVC

- Controller
  - Recebe as entradas
  - Faz chamadas aos modelos de objetos para obter as respostas
    - Aceita as entradas do usuário
    - Instrui o modelo a realizar as ações
    - Faz a interface exibir os resultados baseados naquela entrada
  - Em Java, os Controllers são os Listeners na estrutura de eventos Java

MVC

- Escrever um Model
  - extends java.util.Observable
  - setChanged()
    - Sinaliza que o estado do modelo foi alterado
  - notifyObservers()
    - Envia uma mensagem de alteração para cada observador registrado (View)

MVC

- Model:

```
public class TemperatureModel extends java.util.Observable {
	private double temperatureC = 0.0;

	public double getC(){return temperatureC;}

	public void setC(double tempC) {
		temperatureC = tempC;
		setChanged();
		notifyObservers();
	}
}
```

- Extende um objeto “observável”

MVC

- View:
  - Definição básica do Layout

  - implements java.util.Observer
    - Irá “observar” o modelo do slide anterior
      - TemperaturaModel

MVC

- Método obrigatório ao implementar java.util.Observer

```
public void update(Observable t, Object o) {
      display.setText("" + model.getC());
}
```

  - Atualiza a interface quando o objeto “observável” sinalizar alguma alteração em seu estado

MVC

- Controller

```
public class TemperatureController implements ActionListener {
	private TemperatureModel model;

	public TemperatureController(TemperatureModel m) {
		model = m;
	}

	public void actionPerformed(ActionEvent e) {
		JButton source = (JButton) e.getSource();
		if (source.getText()=="Aumentar") {
			model.setC(model.getC()+1);
		} else {
			model.setC(model.getC()-1);
		}
	}
}
```

MVC

- Linkar o Model e o Controller na View

```
model.addObserver(this);
upButton.addActionListener(new TemperatureController(model));
downButton.addActionListener(new TemperatureController(model));
```

1. Linkar o observador ao objeto observado
    1. View ao Model
2. Linkar o Controller aos componentes da View