In [2]:

```
import numpy as np
from sklearn.datasets import fetch_lfw_people
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.layers import BatchNormalization, Dropout
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_lfw_people
from sklearn.preprocessing import OneHotEncoder
from tensorflow.keras.applications import VGG16
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import img_to_array, array_to_img, l
from tensorflow.keras.applications.vgg16 import preprocess_input
from PIL import Image
from keras.models import load_model
from sklearn.metrics.pairwise import cosine_similarity
import warnings
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
```

In [3]:

```
lfw_people = fetch_lfw_people(min_faces_per_person=70, resize=0.4)

images = lfw_people.images
target_names = lfw_people.target_names
X = lfw_people.data
y = lfw_people.target
n_classes = target_names.shape[0]

n_samples, h, w = lfw_people.images.shape
X_resized = np.zeros((n_samples, 224, 224, 3))

for i in range(n_samples):
    img = X[i].reshape(h, w)
    img = Image.fromarray(np.uint8(img * 200))  # Умножаем на 255 для преобраз
    img = img.convert('RGB')  # Конвертируем в RGB
    img = img.resize((224, 224))  # Изменяем размер
    X_resized[i] = img_to_array(img)  #

# Нормализация изображений
X_resized = preprocess_input(X_resized)

encoder = OneHotEncoder()
y_onehot = encoder.fit_transform(y.reshape(-1, 1))


X_train, X_test, y_train, y_test = train_test_split(X_resized, y, test_size=0.


# Выведем несколько
n_images = 7
```

```
plt.figure(figsize=(15, 8))

for i in range(n_images):
    plt.subplot(1, n_images, i + 1)
    plt.imshow(array_to_img(X_test[i]))
    plt.title(f"{y_test[i]}: {target_names[y_test[i]]}")
    plt.axis('off')

plt.show()
```

In [5]:

```
datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

datagen.fit(X_train)

train_generator = datagen.flow(X_train, y_train, batch_size=32)
validation_generator = datagen.flow(X_test, y_test, batch_size=32)


base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 22
for layer in base_model.layers:
    layer.trainable = False

x = base_model.output
x = Flatten()(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(n_classes, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metric

# Обучение модели
with warnings.catch_warnings():
    warnings.simplefilter("ignore")
    model.fit(train_generator,
          steps_per_epoch=len(X_train) // 32,
          epochs=10,
          validation_data=validation_generator,
          validation_steps=len(X_test) // 32)
```

```
Epoch 1/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 72s 2s/step - accuracy: 0.3070 - loss: 15.2907 - va
l_accuracy: 0.4883 - val_loss: 1.4379
Epoch 2/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 16s 474ms/step - accuracy: 0.4062 - loss: 1.5593 - v
al_accuracy: 0.5625 - val_loss: 1.3861
Epoch 3/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 73s 2s/step - accuracy: 0.4090 - loss: 1.8544 - va
l_accuracy: 0.5781 - val_loss: 1.3367
Epoch 4/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 17s 482ms/step - accuracy: 0.3125 - loss: 1.5138 - v
al_accuracy: 0.5273 - val_loss: 1.3298
Epoch 5/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 74s 2s/step - accuracy: 0.4484 - loss: 1.5753 - va
l_accuracy: 0.5391 - val_loss: 1.1801
Epoch 6/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 17s 488ms/step - accuracy: 0.6562 - loss: 1.2048 - v
al_accuracy: 0.5938 - val_loss: 1.1497
Epoch 7/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 74s 2s/step - accuracy: 0.4684 - loss: 1.4454 - va
l_accuracy: 0.6367 - val_loss: 1.1018
Epoch 8/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 17s 487ms/step - accuracy: 0.6875 - loss: 1.1793 - v
al_accuracy: 0.6328 - val_loss: 1.1238
Epoch 9/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 74s 2s/step - accuracy: 0.5182 - loss: 1.4447 - va
l_accuracy: 0.6016 - val_loss: 1.0801
Epoch 10/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 17s 486ms/step - accuracy: 0.5625 - loss: 1.1545 - v
al_accuracy: 0.6367 - val_loss: 1.0561
```

In [6]:

```python
from sklearn.model_selection import train_test_split

loss, accuracy = model.evaluate(X_test, y_test)
print(f'Loss: {loss}, Accuracy: {accuracy}')
```

```
9/9 ━━━━━━━━━━━━━━━━━━━━ 15s 2s/step - accuracy: 0.7041 - loss: 0.8319
Loss: 0.8931705355644226, Accuracy: 0.682170569896698
```

In [7]:

```python
model.save('face_recognition_model_v8.keras')
```

Тест

In [8]:

```python
i = 8 # выбор картинки из тестовой выборки
```

In [9]:

```python
plt.figure(figsize=(7, 3))
plt.imshow(array_to_img(X_test[i]))
plt.title(f"{y_test[i]}: {target_names[y_test[i]]}")
plt.axis('off')
plt.show()
```

In [10]:

```
prediction = model.predict(np.expand_dims(X_test[i].reshape(224, 224, 3), axis
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 224ms/step
```

In [11]:

```
print(f"распознан класс: {np.argmax(prediction)}")
```

```
распознан класс: 3
```

Тестирование с моими картинками

In [12]:

```
model = load_model('face_recognition_model_v8.keras')
```

In [22]:

```
def get_embedding(img_path):
    img = image.load_img(img_path, target_size=(224, 224))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array = preprocess_input(img_array)
    embedding = model.predict(img_array)
    return embedding

your_image_embedding = get_embedding("vlada_images/1.png")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
```

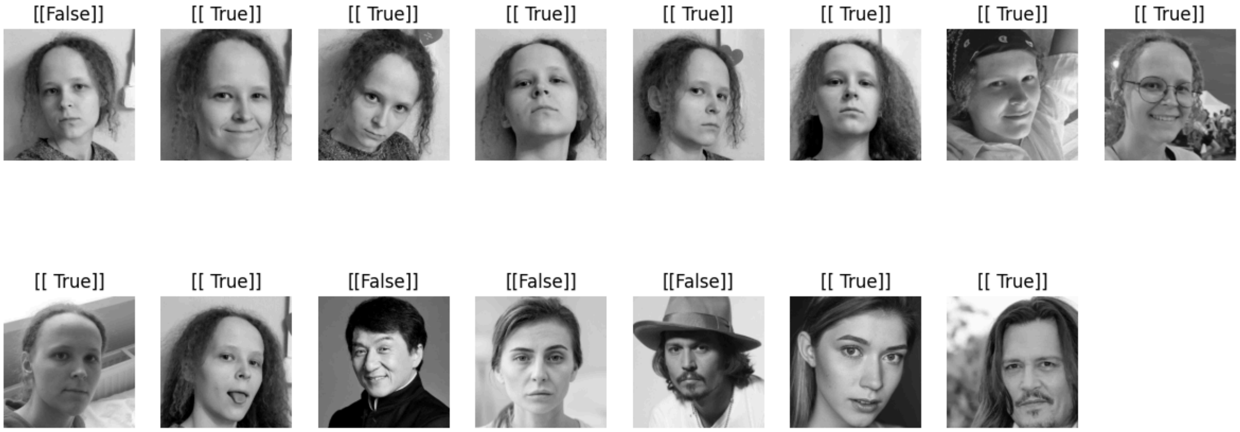
In [23]:

```
def is_not_you(your_embedding, test_embedding, threshold=0.5):
    similarity = cosine_similarity(your_embedding, test_embedding)
    return similarity < threshold

plt.figure(figsize=(15, 6))
N = 16
s_me = 0
s_not_me = 0
for i in range(1, N):
    file_name = f"vlada_images/{i}.png"
    test_embedding = get_embedding(file_name)
    result = is_not_you(your_image_embedding, test_embedding)

    plt.subplot(2, int(N/2), i)
    plt.imshow(Image.open(file_name))
    plt.title(f"{result}")
    plt.axis('off')
    if i <= 10 and result:
        s_me += 1
    if i > 10 and result:
        s_not_me += 1
plt.show()
print(f"Итого \n меня распознано {s_me}/10, \n ошибочно меня распознано {s_not
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 124ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 126ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 125ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 118ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 123ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 128ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 124ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 122ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 126ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 126ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 118ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 120ms/step
```



```
Итого 
 меня распознано 9/10, 
 ошибочно меня распознано 2/5
```

```
In [ ]:
```