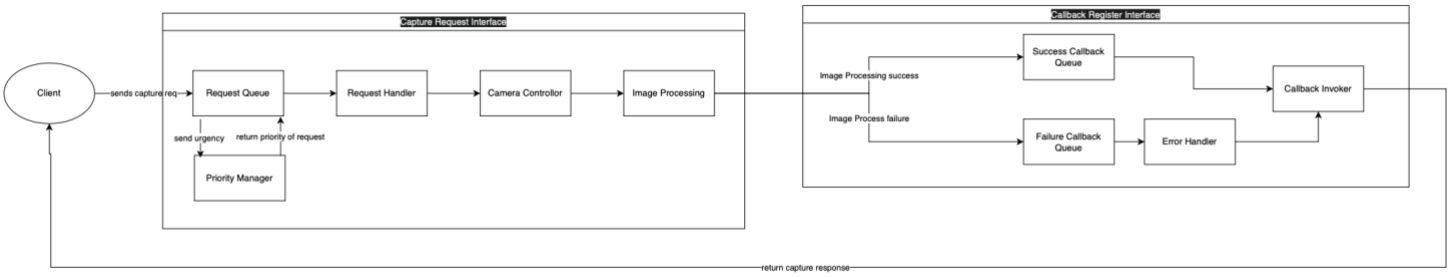
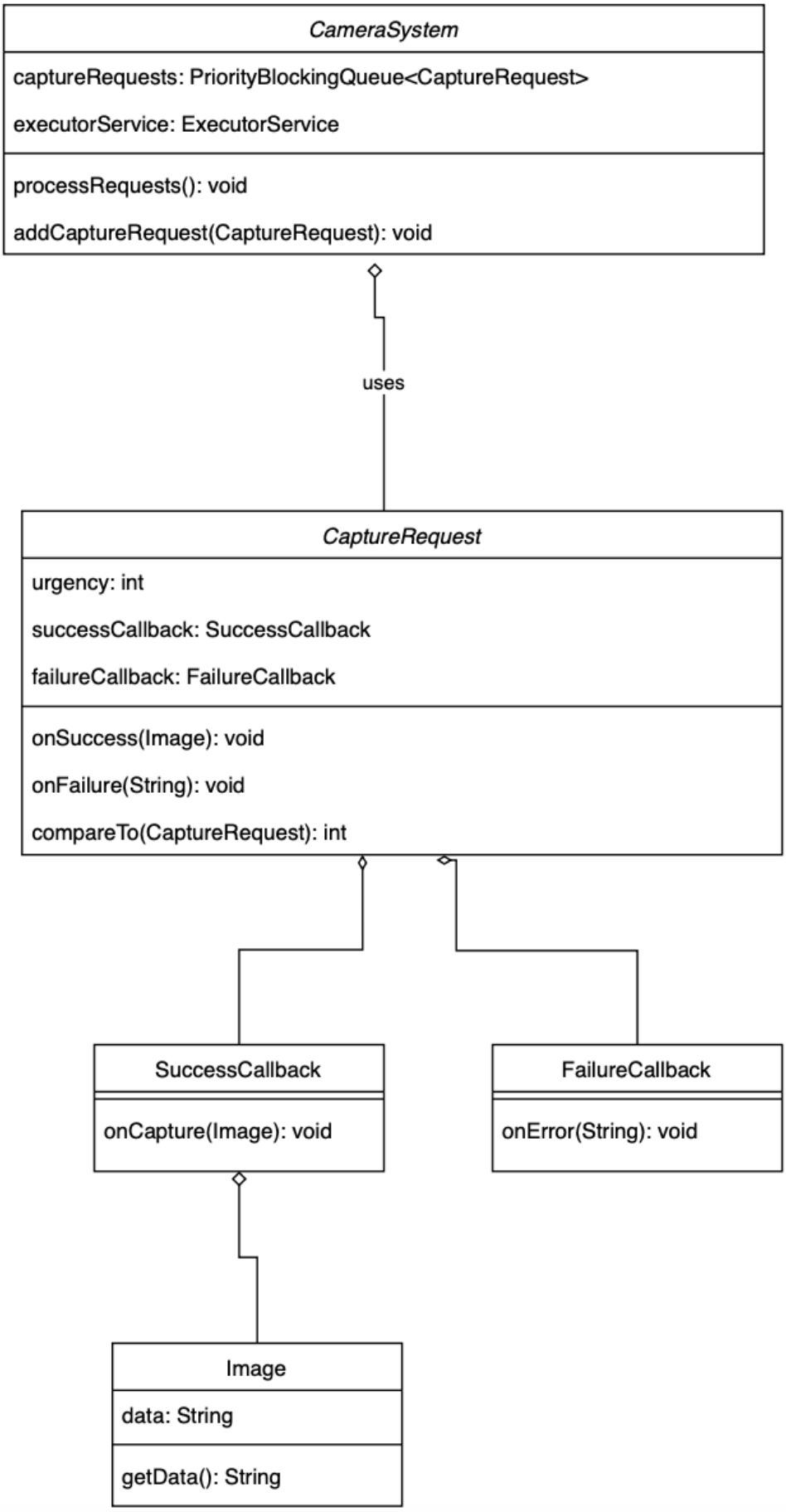
HLD:



1) **Client:** This is the client component which sends capture requests and expects response
2) **Request Queue:** This will be a priority queue where capture requests from clients will be stored
3) **Priority Manager:** . The priority will be set based on the urgency of the request by this component
4) **Request Handler:** This component will fetch requests from the Request Queue and process them. It will manage the multithreading and synchronization of requests and callbacks
5) **Camera Controller**: This component will interact with the physical camera system to capture images
6) **Image Processor**: This component will process the raw images captured by the Camera Controller
7) **Success Callback Queue**: This queue will store success callbacks from the clients
8) **Failure Callback Queue**: This queue will store failure callbacks from the clients
9) **Callback Invoker**: This component will invoke the appropriate callback
10) **Error Handler**: This component will handle any errors encountered during the image capture or processing. It will generate error messages to be passed to the failure callbacks.

LLD:


CameraSystem
captureRequests: PriorityBlockingQueue<CaptureRequest>
executorService: ExecutorService
processRequests(): void
addCaptureRequest(CaptureRequest): void
uses
CaptureRequest
urgency: int
successCallback: SuccessCallback
failureCallback: FailureCallback
onSuccess(Image): void
onFailure(String): void
compareTo(CaptureRequest): int
SuccessCallback
onCapture(Image): void
FailureCallback
onError(String): void
Image
data: String
getData(): String

Implementation of CameraSystem:

1) SuccessCallback Interface

```
public interface SuccessCallback {
    void onCapture(Image image);
}
```

2) FailureCallback Interface:

```
public interface FailureCallback {
    void onError(String message);
}
```

3) Image Class:

```
public static class Image {
    private final String data;

    public Image(String data) {
        this.data = data;
    }

    public String getData() {
        return data;
    }
}
```

4) CaptureRequest Class:

```
public static class CaptureRequest implements Comparable<CaptureRequest> {
    private final int urgency;
    private final SuccessCallback successCallback;
    private final FailureCallback failureCallback;

    public CaptureRequest(int urgency, SuccessCallback successCallback,
FailureCallback failureCallback) {
        this.urgency = urgency;
        this.successCallback = successCallback;
        this.failureCallback = failureCallback;
    }

    public int getUrgency() {
        return urgency;
    }

    public void onSuccess(Image image) {
        successCallback.onCapture(image);
    }

    public void onFailure(String message) {
        failureCallback.onError(message);
    }

    @Override
    public int compareTo(CaptureRequest o) {
        return Integer.compare(o.getUrgency(), this.getUrgency());
    }
}
```

5) CameraSystem Class:

```
public class CameraSystem2 {
    private final PriorityBlockingQueue<CaptureRequest> captureRequests = new
PriorityBlockingQueue<>();
    private final ExecutorService executorService =
Executors.newCachedThreadPool();

    public void processRequests() {
        while(true) {
            try {
                CaptureRequest request = captureRequests.take();
                executorService.submit(() -> {
                    try {
                        Thread.sleep(1000);
                        request.onSuccess(new Image("Captured image data"));
                    } catch(InterruptedException e) {
                        e.printStackTrace();
                    }
                });
            } catch(InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    public synchronized void addCaptureRequest(CaptureRequest request) {
        captureRequests.add(request);
    }

    public static void main(String[] args) {
        // Implementation of main method.
    }
}
```

Execution:

```
public static void main(String[] args) {
    CameraSystem2 cameraSystem = new CameraSystem2();

        new Thread(cameraSystem::processRequests).start();

    // Client 1
    cameraSystem.addCaptureRequest(new CaptureRequest(1,
            image -> System.out.println("Client 1 success callback: " +
image.getData()),
            error -> System.out.println("Client 1 failure callback: " + error)
    ));

    // Client 2
    cameraSystem.addCaptureRequest(new CaptureRequest(2,
            image -> System.out.println("Client 2 success callback: " +
image.getData()),
            error -> System.out.println("Client 2 failure callback: " + error)
    ));

    // Client 3
    cameraSystem.addCaptureRequest(new CaptureRequest(3,
            image -> System.out.println("Client 3 success callback: " +
image.getData()),
            error -> System.out.println("Client 3 failure callback: " + error)
    ));

    // Client 4
    cameraSystem.addCaptureRequest(new CaptureRequest(4,
            image -> System.out.println("Client 4 success callback: " +
image.getData()),
            error -> System.out.println("Client 4 failure callback: " + error)
    ));

    // Client 5
    cameraSystem.addCaptureRequest(new CaptureRequest(5,
            image -> System.out.println("Client 5 success callback: " +
image.getData()),
            error -> System.out.println("Client 5 failure callback: " + error)
    ));
}
```

Output: