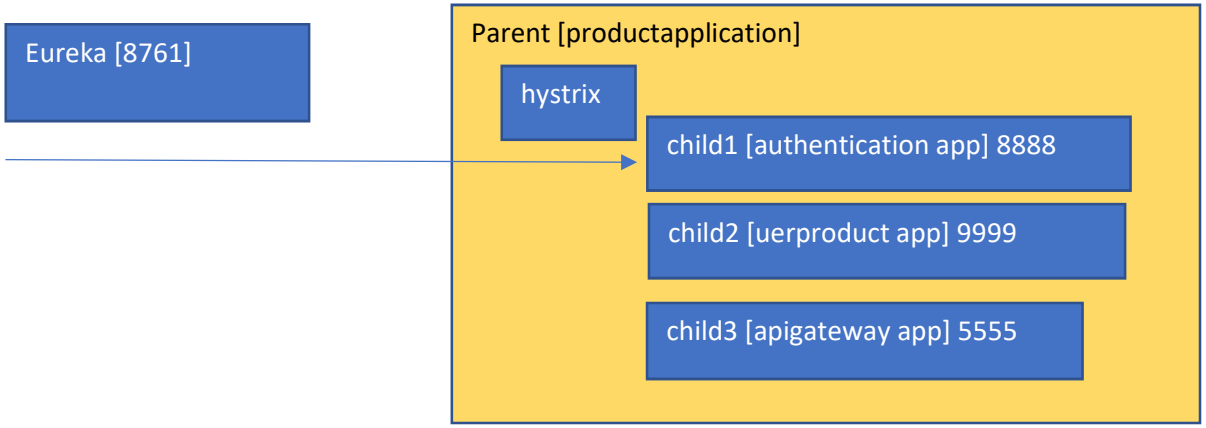
Example.java - Notepad

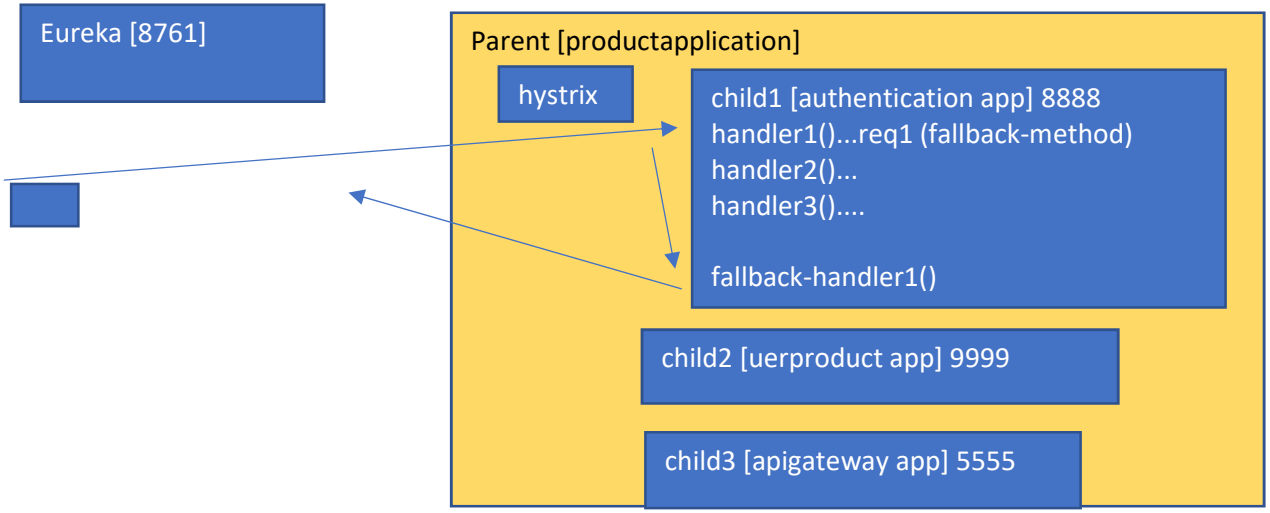
File Edit Format View Help

```
class Example {
        public static void main(String args[]) {
                int i;
                for(i=1;i<=10;i++){
                        System.out.print("  " + i);
                        try{
                                Thread.sleep(1000);
                        }
                        catch(Exception ex) {
                                ex.printStackTrace();
                        }
                }
        }
}
```

```
C:\Users\Babji\Desktop>javac Example.java

C:\Users\Babji\Desktop>java Example
  1  2  3  4  5  6  7  8  9  10
C:\Users\Babji\Desktop>
```





Steps to enable Hystrix on authentication application
and enable fallback method for particular request handler method

Step 1 Makesure authentication application is ready

Step 2 add depedency to authentication application pom

```
<!-- https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-netflix-hystrix -->
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
    <version>2.2.10.RELEASE</version>
</dependency>
```

Step 3 Enable Hystrix at application level

```
1 usage
@SpringBootApplication
@EnableEurekaClient
@EnableHystrix
public class AuthappApplication {

    public static void main(String[] args)

}
```

Step 4 Enable fallback for request handler method in controller
if logincheck() takes morethan X amount of time, then fallback method should run

```
    // if logincheck() takes morethan x amount of time to process -> call fallback method
    @PostMapping("/authenticate")
    @HystrixCommand(fallbackMethod = "fallbackLoginCheck",commandKey = "loginkey", groupKey = "login")
    @HystrixProperty(name="execution.isolation.thread.timeoutInMilliseconds", value="1000")
    public ResponseEntity<?> loginCheck(@RequestBody User user){
        User result = userService.loginCheck(user.getUserId(), user.getPassword());
        if(result!=null){
            Map<String, String> key = securityTokenGenerator.generateToken(result);
            return new ResponseEntity<>(key,HttpStatus.OK);
        }
        else{
            return new ResponseEntity<>( body: "Authentication failed",HttpStatus.NOT_FOUND);
        }
    }
```

Step 5 define fallback method

```
    // fallback method
    public ResponseEntity<?> fallbackLoginCheck(@RequestBody User user){
        String message = "Login service is busy, please try after sometime";
        return new ResponseEntity<>(message,HttpStatus.GATEWAY_TIMEOUT);
    }
```

Solutions if timeout doesnot work

```
35      //http://localhost:8888/auth-app/v1/authenticate [POST]
36      @PostMapping("/authenticate")
37      @HystrixCommand(fallbackMethod = "authenticateFallBack",commandKey = "loginKey",groupKey = "login")
38      @HystrixProperty(name = "execution.isolation.thread.timeoutInMilliseconds",value = "2_000")
39      public ResponseEntity<?>authenticate(@RequestBody User user){
```

```
@HystrixCommand(fallbackMethod = "fallbackLoginCheck",
commandProperties = {
        @HystrixProperty(name =
"execution.isolation.thread.timeoutInMilliseconds", value = "5000")
   })
```