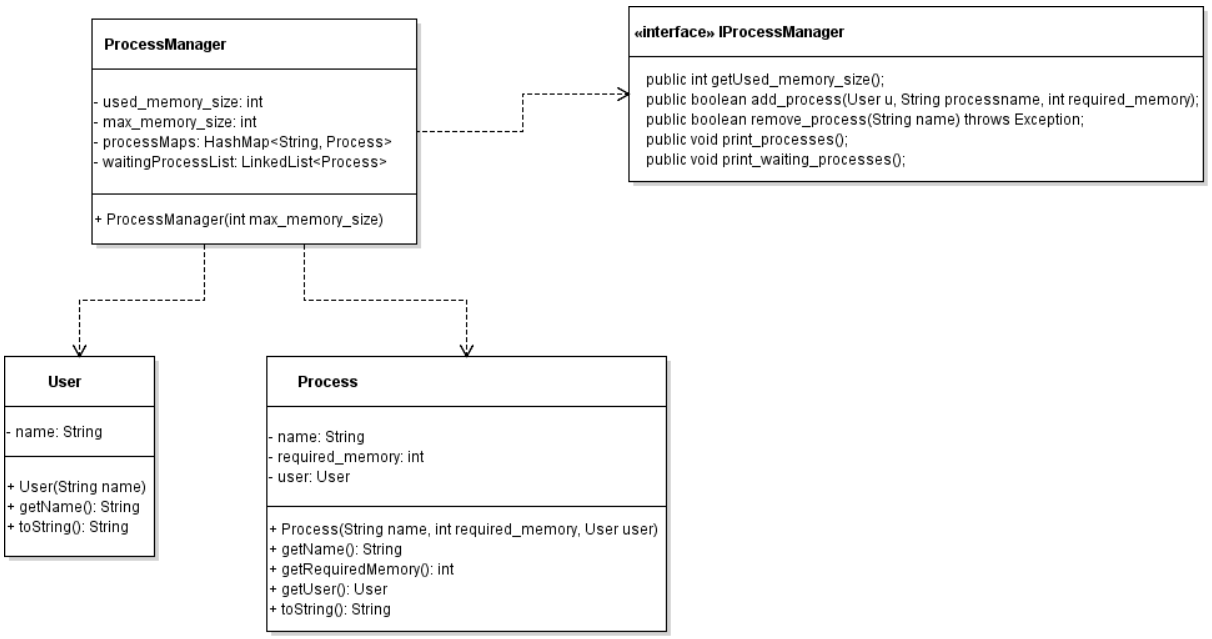

Bir process yönetimi uygulaması yazmanız beklenmektedir.
Programın UML sınıf diyagramı, main sınıfı ve çıktıları paylaşılmıştır.
User, Process, ProcessManager sınıflarını ve IProcessManager interfaceini yazmanız beklenmektedir.
Programda her bir process hafızada belirli bir alan kaplamaktadır. **add_process()** metodu processing ismini, gerekli hafızasını ve process'i başlatan kullanıcıyı alır. Eğer process için yeterli hafıza mevcutsa process **processMaps** HaspMap'ine eklenir. Eğer yeterli hafıza yoksa process **waitingProcessList** e eklenir. **print_processes() methodu processMaps**'de tutulan çalışan processleri yazdırır. **print_waiting_processes() methodu** çalışmak için bekletilen processleri yazdırır. **remove_process()** methodu silinmek istenilen processin ismini alarak ilgili processi çalışan process'lerden siler. Eğer silinmek istenilen process ismi çalışan processler arasında yoksa method "Process cannot found!" mesajı ile bir **Exception** fırlatır. Çalışan process silindikten sonra, bekleyen processler arasından sırasıyla yeterli hafızaya alınabilen processler çalışan processlere eklenmelidir.
getUsed_memory_size() şu anda kullanılmakta olan processlerin toplam hafızasını döndürür.
UML diyagramında bulunmayan method ekleyip kullanabilirsiniz.
Başlangıç Saati: 16:15
Bitiş Saati:17:15
Kodlarınızı yüklemek için form linki: https://forms.gle/yb5RWxnSqnjFmxpf6

Main Sınıfı

```java
public class Main {

    public static void main(String[] args) {
        ProcessManager PM = new ProcessManager(10);
        User u1 = new User("u1");
        User u2 = new User("u2");
        User u3 = new User("u3");

        PM.add_process(u1, "Process1", 2);
        PM.add_process(u1, "Process4", 2);
        PM.add_process(u2, "Process2", 6);



        PM.add_process(u3, "Process3", 3);
        PM.add_process(u1, "Process6", 7);
        PM.add_process(u2, "Process2", 4); // Aynı isimde process olduğu için iki listeyede eklemez. Process isimleri unique olmalı.
        PM.add_process(u2, "Process5", 1);

        System.out.println("1-Process -----------------");
        PM.print_processes();
        System.out.println("1-Waiting Process -----------------");
        PM.print_waiting_processes();

        try {
            PM.remove_process("Process2"); // Catch'e girmez çünkü process mevcut.
        }
        catch (Exception ex) {
            System.out.println(ex);
        }

        System.out.println("After Process2 Removed");

        System.out.println("2-Process -----------------");
        PM.print_processes();
        System.out.println("2-Waiting Process -----------------");
        PM.print_waiting_processes();

        System.out.println(PM.getUsed_memory_size());

        try {
            PM.remove_process("OlmayanProcess"); // Catch'e girer process mevcut değil.
        }
        catch (Exception ex) {
            System.out.println(ex);
        }

    }

}
```

Konsol Çıktısı

```
1-Process -----------------
Process2 -> Process{name=Process2, required_memory=6, user=User{name=u2}}
Process1 -> Process{name=Process1, required_memory=2, user=User{name=u1}}
Process4 -> Process{name=Process4, required_memory=2, user=User{name=u1}}
1-Waiting Process -----------------
Process{name=Process3, required_memory=3, user=User{name=u3}}
Process{name=Process6, required_memory=7, user=User{name=u1}}
Process{name=Process5, required_memory=1, user=User{name=u2}}
After Process2 Removed
2-Process -----------------
Process1 -> Process{name=Process1, required_memory=2, user=User{name=u1}}
Process4 -> Process{name=Process4, required_memory=2, user=User{name=u1}}
Process3 -> Process{name=Process3, required_memory=3, user=User{name=u3}}
Process5 -> Process{name=Process5, required_memory=1, user=User{name=u2}}
2-Waiting Process -----------------
Process{name=Process6, required_memory=7, user=User{name=u1}}
8
java.lang.Exception: Process cannot found!
BUILD SUCCESSFUL (total time: 0 seconds)
```