МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО

ФАИС

Кафедра «Информатика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6
по дисциплине «**Операционные системы и среды**»

на тему: «**Планирование процессов**»

Выполнил: студент гр. ИП-32
Прокопенко А. Р.
Принял: преподаватель
Процкая М. А.

Дата сдачи отчета: _____
Дата допуска к защите: _____
Дата защиты: _____

Гомель 2022

Цель: изучить типовые алгоритмы планирования процессов.

Задание

Вариант	Продолжительности процессов	Время появления в очереди	Приоритеты процессов
10	P0 – 7; P1 – 1; P2 – 4; P3 – 6;	P0 – 1; P1 – 2; P2 – 4; P3 – 0;	P0 – 2; P1 – 1; P2 – 3; P3 – 4;

1. Не вытесняющие алгоритмы планирования процессов.
Выполнить различные алгоритмы планирований – First-Come, First-Served (FCFS) (прямой и обратный), Round Robin (RR), Shortest-Job-First (SJF) (не вытесняющий), Shortest-Job-First (SJF) (не вытесняющий приоритетный) для данных приведенных в таблице 2.1 в соответствии со своим вариантом (номер по журналу). Вычислить полное время выполнения все процессов и каждого в отдельности, время ожидание для каждого процесса. Рассчитать среднее время выполнения процесса и среднее время ожидания. Результаты оформить в виде таблиц иллюстрирующих работу процессов.

2. Вытесняющие алгоритмы планирования процессов.
Выполнить различные алгоритмы планирований – Shortest-Job-First (SJF) (вытесняющий) и Shortest-Job-First (SJF) (приоритетный) для данных приведенных в таблице 2.1 в соответствии со своим вариантом. Вычислить полное время выполнения все процессов и каждого в отдельности, время ожидание для каждого процесса. Рассчитать среднее время выполнения процесса и среднее время ожидания. Результаты оформить в виде таблиц иллюстрирующих работу процессов.

3. Программная реализация алгоритмов задания 1 и 2.
Разработать программную реализацию алгоритмов задания 1 и 2.

Выполнение
1. Не вытесняющие алгоритмы планирования процессов.
First-Come, First-Served (FCFS) (прямой):

Процессы	Время ожидания	Полное время выполнения	Время появления в очереди
P0	5	7	1
P1	11	1	2
P2	10	4	4
P3	0	6	0

Время	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
P0		Г	Г	Г	Г	Г	И	И	И	И	И	И	И					
P1			Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	И				
P2					Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	И	И	И	И
P3	И	И	И	И	И	И												

«И» - исполняется, «Г» - в ожидании готовности, « » - завершен или не начат.
Полное время выполнения всех процессов: 7 + 1 + 4 + 6 = 18.
Среднее время выполнения процесса: 18 / 4 = 4.5.
Среднее время ожидания процесса: (5 + 11 + 10 + 0) / 4 = 6.5.

First-Come, First-Served (FCFS) (обратный):

Процессы	Время ожидания	Полное время выполнения	Время появления в очереди
P0	3	7	2
P1	3	1	1
P2	0	4	0
P3	8	6	4

Время	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
P0			Г	Г	Г	И	И	И	И	И	И	И						
P1		Г	Г	Г	И													
P2	И	И	И	И														
P3					Г	Г	Г	Г	Г	Г	Г	Г	И	И	И	И	И	И

«И» - исполняется, «Г» - в ожидании готовности, « » - завершен или не начат.
Полное время выполнения всех процессов: 7 + 1 + 4 + 6 = 18.
Среднее время выполнения процесса: 18 / 4 = 4.5.
Среднее время ожидания процесса: (3 + 3 + 0 + 8) / 4 = 3.5.

Round Robin (RR):

Процессы	Время ожидания	Полное время выполнения	Время появления в очереди
P0	10	7	1
P1	4	1	2
P2	9	4	4
P3	7	6	0

Время	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
P0		Г	Г	И	И	И	Г	Г	Г	Г	Г	Г	Г	И	И	И	Г	И
P1			Г	Г	Г	Г	И											
P2					Г	Г	Г	И	И	И	Г	Г	Г	Г	Г	Г	И	
P3	И	И	И	Г	Г	Г	Г	Г	Г	Г	И	И	И					

Пусть квант времени равняется 3.
«И» - исполняется, «Г» - в ожидании готовности, « » - завершен или не начат.
Полное время выполнения всех процессов: 7 + 1 + 4 + 6 = 18.
Среднее время выполнения процесса: 18 / 4 = 4.5.
Среднее время ожидания процесса: (10 + 4 + 9 + 7) / 4 = 7.5.

Shortest-Job-First (SJF) (не вытесняющий):

Процессы	Время ожидания	Полное время выполнения	Время появления в очереди
P0	10	7	1
P1	4	1	2
P2	3	4	4
P3	0	6	0

Время	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
P0		Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	И	И	И	И	И	И	И
P1			Г	Г	Г	Г	И											
P2					Г	Г	Г	И	И	И	И							
P3	И	И	И	И	И	И												

«И» - исполняется, «Г» - в ожидании готовности, « » - завершен или не начат.
Полное время выполнения всех процессов: 7 + 1 + 4 + 6 = 18.
Среднее время выполнения процесса: 18 / 4 = 4.5.
Среднее время ожидания процесса: (10 + 4 + 3 + 0) / 4 = 4.25.

Shortest-Job-First (SJF) (не вытесняющий приоритетный):

Процессы	Время ожидания	Полное время выполнения	Время появления в очереди	Приоритет
P0	6	7	1	2
P1	4	1	2	1
P2	10	4	4	3
P3	0	6	0	4

Время	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
P0		Г	Г	Г	Г	Г	Г	И	И	И	И	И	И	И				
P1			Г	Г	Г	Г	И											
P2					Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	И	И	И	И
P3	И	И	И	И	И	И												

Будем считать, что большее значение приоритета соответстует меньшему приоритету, т.е. наиболее приоритетным в данном примере является P1, а наименее приоритетным — процесс P3.
«И» - исполняется, «Г» - в ожидании готовности, « » - завершен или не начат.
Полное время выполнения всех процессов: 7 + 1 + 4 + 6 = 18.
Среднее время выполнения процесса: 18 / 4 = 4.5.
Среднее время ожидания процесса: (10 + 6 + 4 + 0) / 4 = 5.

2. Вытесняющие алгоритмы планирования процессов.

Shortest-Job-First (SJF) (вытесняющий):

Процессы	Время ожидания	Полное время выполнения	Время появления в очереди
P0	10	7	1
P1	0	1	2
P2	3	4	4
P3	1	6	0

Время	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
P0		Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	И	И	И	И	И	И	И
P1			И															
P2					Г	Г	Г	И	И	И	И							
P3	И	И	Г	И	И	И	И											

«И» - исполняется, «Г» - в ожидании готовности, « » - завершен или не начат.
Полное время выполнения всех процессов: 7 + 1 + 4 + 6 = 18.
Среднее время выполнения процесса: 18 / 4 = 4.5.
Среднее время ожидания процесса: (10 + 0 + 3 + 1) / 4 = 3.5.

Shortest-Job-First (SJF) (приоритетный):

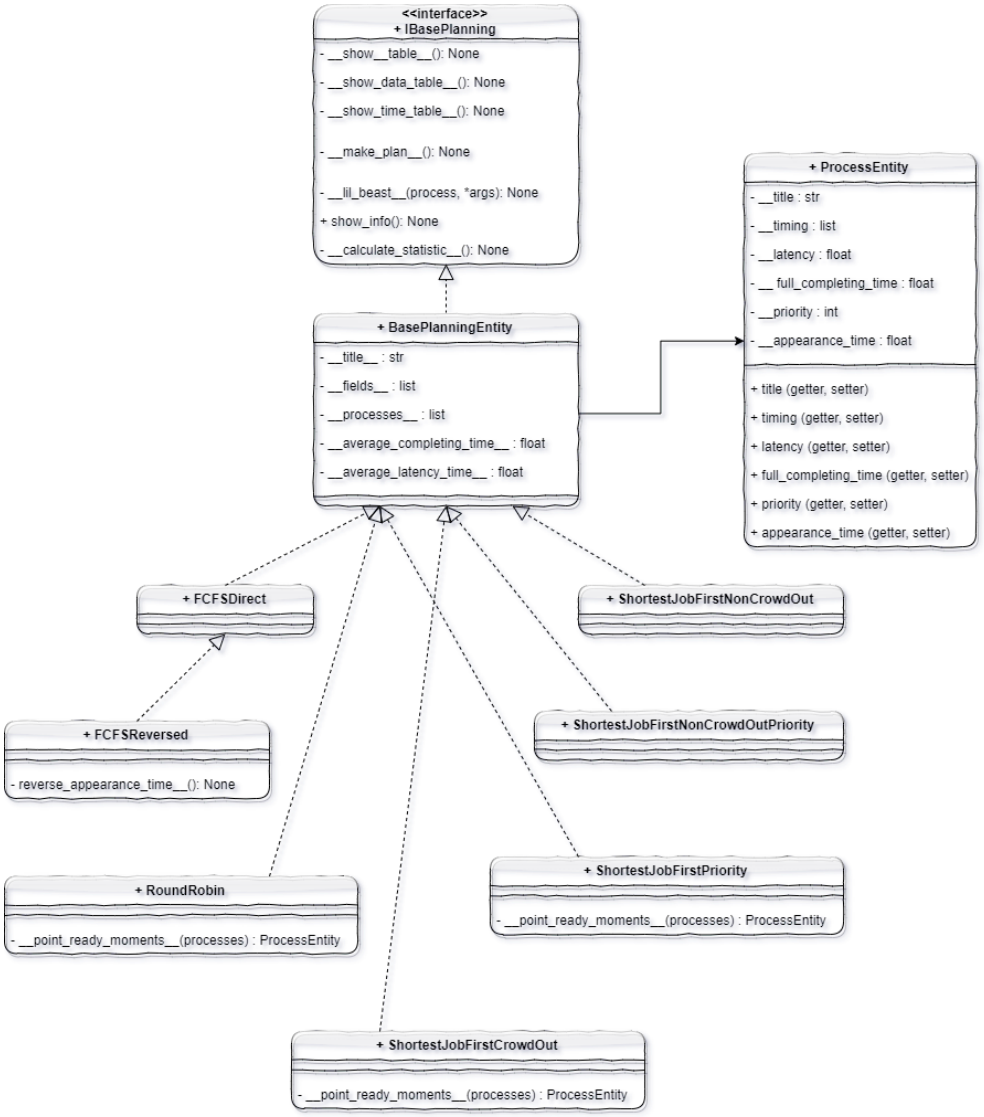
Процессы	Время ожидания	Полное время выполнения	Время появления в очереди	Приоритет
P0	1	7	1	2
P1	0	1	2	1
P2	5	4	4	3
P3	12	6	0	4

Время	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
P0		И	Г	И	И	И	И	И	И									
P1			И															
P2					Г	Г	Г	Г	Г	И	И	И	И					
P3	И	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	И	И	И	И	И

Будем считать, что большее значение приоритета соответстует меньшему приоритету, т.е. наиболее приоритетным в данном примере является P1, а наименее приоритетным — процесс P3.
«И» - исполняется, «Г» - в ожидании готовности, « » - завершен или не начат.
Полное время выполнения всех процессов: 7 + 1 + 4 + 6 = 18.
Среднее время выполнения процесса: 18 / 4 = 4.5.
Среднее время ожидания процесса: (12 + 0 + 5 + 1) / 4 = 4.5.

В данном случае самыми эффективным оказались First-Come, First-Served (FCFS) (обратный) и Shortest-Job-First (SJF) (вытесняющий).

3. Программная реализация алгоритмов задания 1 и 2.
Диаграмма классов программы:



Код программы на ЯП Python:

IbasePlanning:

```
import abc


class IBasePlanning(abc.ABC):

    @abc.abstractmethod
    def __show_tables__(self) -> None:
        pass

    @abc.abstractmethod
    def __show_data_table__(self) -> None:
        pass

    @abc.abstractmethod
    def __show_time_table__(self) -> None:
        pass

    @abc.abstractmethod
    def __make_plan__(self) -> None:
        pass

    @abc.abstractmethod
    def __lil_beast__(self, process, *args):
        """
        This creature is needed to eat full_completing_time and provide timing with latency.
        Wonderful and cruel it is a part of the great mechanism...
        Laurent...
        """
        pass

    @abc.abstractmethod
    def show_info(self):
        pass

    @abc.abstractmethod
    def __calculate_statistic__(self):
        pass
```

BasePlanningEntity:

```
from PlanningSystems.Interfaces.IBasePlanning import IBasePlanning
from prettytable import PrettyTable


class BasePlanningEntity(IBasePlanning):
    __title__ = None
    __fields__ = ['Processes', 'Latency', 'Full completing time', 'Appearance time', 'Priority']

    def __init__(self, processes: list):
        self.__processes__ = sorted(processes, key=lambda x: x.title)
        self.__whole_time__ = sum(process.full_completing_time for process in self.__processes__)
        self.__average_completing_time__ = None
        self.__average_latency_time__ = None
```

```
    def __show_tables__(self) -> None:
        self.__make_plan__()
        self.__show_data_table__()
        self.__show_time_table__()

    def __show_data_table__(self) -> None:
        table = PrettyTable()
        table.title = self.__title__
        table.field_names = self.__fields__
        data_rows = [
            (process.title, process.latency, process.full_completing_time, process.appearance_time, process.priority)
            for process in self.__processes__
        ]
        table.add_rows(data_rows)
        print(table)

    def __show_time_table__(self) -> None:
        table = PrettyTable()
        table.title = self.__title__
        table.field_names = ['Time'] + [str(i) for i in range(1, self.__whole_time__ + 1)]
        data_rows = [[process.title] + process.timing for process in self.__processes__]
        table.add_rows(data_rows)
        print(table)

    def __make_plan__(self) -> None:
        for process in self.__processes__:
            process.timing = [" " for i in range(self.__whole_time__)]

    def show_info(self):
        self.__show_tables__()
        self.__calculate_statistic__()
        print(f'Full completing time: {self.__whole_time__}')
        print(f'Average completing time: {self.__average_completing_time__}')
        print(f'Average latency time: {self.__average_latency_time__}')
        print('-' * 50, '\n\n')

    def __calculate_statistic__(self):
        self.__average_completing_time__ = sum([process.full_completing_time for process in self.__processes__]) \
                                           / len(self.__processes__)
        self.__average_latency_time__ = sum([process.latency for process in self.__processes__])\
                                        / len(self.__processes__)

    def __lil_beast__(self, process, *args):
        pass
```

ProcessEntity:

```
class ProcessEntity:
    def __init__(self, title, latency, full_completing_time, priority, appearance_time, timing=None):
        self.__title = title
        self.__timing = timing
        self.__latency = latency
        self.__full_completing_time = full_completing_time
        self.__priority = priority
        self.__appearance_time = appearance_time

    @property
    def title(self):
        return self.__title

    @property
    def latency(self):
```

```
        return self.__latency

    @property
    def full_completing_time(self):
        return self.__full_completing_time

    @property
    def priority(self):
        return self.__priority

    @property
    def appearance_time(self):
        return self.__appearance_time

    @property
    def timing(self):
        return self.__timing

    @title.setter
    def title(self, title):
        self.__title = title

    @latency.setter
    def latency(self, latency):
        self.__latency = latency

    @full_completing_time.setter
    def full_completing_time(self, full_completing_time):
        self.__full_completing_time = full_completing_time

    @priority.setter
    def priority(self, priority):
        self.__priority = priority

    @appearance_time.setter
    def appearance_time(self, appearance_time):
        self.__appearance_time = appearance_time

    @timing.setter
    def timing(self, timing):
        self.__timing = timing
```

FCFSDirect:

```
from Entities.BasePlanningEntity import BasePlanningEntity
import copy


class FCFSDirect(BasePlanningEntity):
    __title__ = "FIRST-COME, FIRST-SERVED, direct"

    def __init__(self, processes: list):
        super().__init__(processes)

    def __make_plan__(self) -> None:
        super().__make_plan__()
        # sorted_processes - objects which will lose their full completing time,
        # but they will have latency and timing
        sorted_processes = sorted(copy.deepcopy(self.__processes__), key=lambda x: x.appearance_time)
        start = min(process.appearance_time for process in sorted_processes)
        for i, process in enumerate(sorted_processes, start=0):
            sorted_processes[i] = self.__lil_beast__(process, start, start + process.full_completing_time)
            start += process.full_completing_time
        sorted_processes = sorted(sorted_processes, key=lambda x: x.title)
        for i in range(len(sorted_processes)):
```

```
            self.__processes__[i].timing = sorted_processes[i].timing
            self.__processes__[i].latency = sorted_processes[i].latency

    def __lil_beast__(self, process, *args):
        """
        See description in an abstract class file.
        :param process: the process itself.
        :param args: start and end of timing (in MAIN FLOW)
        :return: sophisticated lil beast itself...
        """
        start, end = args
        process.timing[process.appearance_time:end] = ["Г" for i in range(process.appearance_time, end)]
        process.timing[start:end] = ['И' for i in range(start, end)]
        process.latency = process.timing.count('Г')
        return process
```

FCFSReversed:

```
from PlanningSystems.FCFSDirect import FCFSDirect


class FCFSReversed(FCFSDirect):
    __title__ = "FIRST-COME, FIRST-SERVED, reversed"

    def __init__(self, processes: list):
        super().__init__(processes)
        self.__reverse_appearance_time__()

    def __reverse_appearance_time__(self):
        self.__processes__ = sorted(self.__processes__, key=lambda x: x.appearance_time)
        appearance_time = [process.appearance_time for process in self.__processes__]
        appearance_time.reverse()
        for i in range(len(self.__processes__)):
            self.__processes__[i].appearance_time = appearance_time[i]
        self.__processes__ = sorted(self.__processes__, key=lambda x: x.title)
```

RoundRobin:

```
from Entities.BasePlanningEntity import BasePlanningEntity
import copy


class RoundRobin(BasePlanningEntity):
    __title__ = "ROUND ROBIN (RR)"
    portion = 3

    def __init__(self, processes: list):
        super().__init__(processes)

    def __make_plan__(self) -> None:
        super().__make_plan__()
        sorted_processes = sorted(copy.deepcopy(self.__processes__), key=lambda x: x.appearance_time)
        start = min(process.appearance_time for process in sorted_processes)
        while start < self.__whole_time__:
            for i in range(0, len(sorted_processes)):
                sorted_processes[i], start = self.__lil_beast__(sorted_processes[i], start)
        sorted_processes = self.__point_ready_moments__(sorted_processes)
        sorted_processes = sorted(sorted_processes, key=lambda x: x.title)
        for i in range(len(sorted_processes)):
            self.__processes__[i].timing = sorted_processes[i].timing
            self.__processes__[i].latency = sorted_processes[i].latency
```

```
    def __lil_beast__(self, process, *args):
        """
        See description in an abstract class file.
        :param process: the process itself.
        :param args: current time.
        :return: process, time.
        """
        time = args[0]
        process.timing[time:time + min(self.portion, process.full_completing_time)] \
            = ['И' for i in range(min(self.portion, process.full_completing_time))]
        time += min(self.portion, process.full_completing_time)
        process.full_completing_time -= min(self.portion, process.full_completing_time)
        return process, time

    @staticmethod
    def __point_ready_moments__(processes):
        """
        It should be used after __lil_beast__.
        :param processes: processes
        :return: modified processes
        """
        for i, process in enumerate(processes, start=0):
            start = process.appearance_time
            end = ''.join(process.timing).rfind('И')
            for j in range(start, end + 1):
                if process.timing[j] != 'И':
                    process.timing[j] = 'Г'
                    process.latency += 1
        return processes
```

ShortestJobFirstCrowdOut:

```
from Entities.BasePlanningEntity import BasePlanningEntity
import copy


class ShortestJobFirstCrowdOut(BasePlanningEntity):
    __title__ = "Shortest Job First Crowd Out"

    def __init__(self, processes: list):
        super().__init__(processes)

    def __make_plan__(self) -> None:
        super().__make_plan__()
        processes_copy = copy.deepcopy(self.__processes__)
        time = 0
        while time < self.__whole_time__:
            active_processes = [
                process for process in processes_copy
                if
                process.full_completing_time != 0 and process.appearance_time <= time
            ]
            active_processes.sort(key=lambda x: x.full_completing_time)
            if len(active_processes) > 0:
                active_processes[0] = self.__lil_beast__(active_processes[0], time)
            time += 1
        self.__point_ready_moments__(processes_copy)
        for i, process in enumerate(processes_copy, start=0):
            self.__processes__[i].latency = process.latency
            self.__processes__[i].timing = process.timing
```

```
    def __lil_beast__(self, process, *args):
        """
        Lil beast itself.
        :param process: process
        :param args: time
        :return: updated process
        """
        time = args[0]
        process.timing[time:time + 1] = ['И' for i in range(time, time + 1)]
        process.full_completing_time -= 1
        return process

    @staticmethod
    def __point_ready_moments__(processes):
        """
        It should be used after __lil_beast__.
        :param processes: processes
        :return: modified processes
        """
        for i, process in enumerate(processes, start=0):
            start = process.appearance_time
            end = ''.join(process.timing).rfind('И')
            for j in range(start, end + 1):
                if process.timing[j] != 'И':
                    process.timing[j] = 'Г'
                    process.latency += 1
        return processes
```

ShortestJobFirstNonCrowdOut:

```
from Entities.BasePlanningEntity import BasePlanningEntity
import copy


class ShortestJobFirstNonCrowdOut(BasePlanningEntity):
    __title__ = "Shortest Job First Non Crowd Out"

    def __init__(self, processes: list):
        super().__init__(processes)

    def __make_plan__(self) -> None:
        super().__make_plan__()
        processes_copy = copy.deepcopy(self.__processes__)
        time = 0
        while time < self.__whole_time__:
            active_processes = [
                process for process in processes_copy
                if
                process.full_completing_time != 0 and process.appearance_time <= time
            ]
            active_processes.sort(key=lambda x: x.full_completing_time)
            if len(active_processes) > 0:
                active_processes[0], time = self.__lil_beast__(active_processes[0], time)
            else:
                time += 1
        for i, process in enumerate(processes_copy, start=0):
            self.__processes__[i].latency = process.latency
            self.__processes__[i].timing = process.timing

    def __lil_beast__(self, process, *args):
        """
        Lil beast itself.
```

```
        :param process: process
        :param args: time
        :return: updated process, time
        """
        time = args[0]
        process.timing[process.appearance_time:time + process.full_completing_time] = \
            ['Г' for i in range(process.appearance_time, time + process.full_completing_time)]
        process.timing[time:time + process.full_completing_time] = \
            ['И' for i in range(time, time + process.full_completing_time)]
        process.latency = process.timing.count('Г')
        time += process.full_completing_time
        process.full_completing_time = 0
        return process, time
```

ShortestJobFirstNonCrowdOutPriority:

```
from Entities.BasePlanningEntity import BasePlanningEntity
import copy


class ShortestJobFirstNonCrowdOutPriority(BasePlanningEntity):
    __title__ = "Shortest Job First Non Crowd Out Priority"

    def __init__(self, processes: list):
        super().__init__(processes)

    def __make_plan__(self) -> None:
        super().__make_plan__()
        processes_copy = copy.deepcopy(self.__processes__)
        time = 0
        while time < self.__whole_time__:
            active_processes = [
                process for process in processes_copy
                if
                process.full_completing_time != 0 and process.appearance_time <= time
            ]
            active_processes.sort(key=lambda x: x.priority)
            if len(active_processes) > 0:
                active_processes[0], time = self.__lil_beast__(active_processes[0], time)
            else:
                time += 1
        for i, process in enumerate(processes_copy, start=0):
            self.__processes__[i].latency = process.latency
            self.__processes__[i].timing = process.timing

    def __lil_beast__(self, process, *args):
        """
        Lil beast itself.
        :param process: process
        :param args: time
        :return: updated process, time
        """
        time = args[0]
        process.timing[process.appearance_time:time + process.full_completing_time] = \
            ['Г' for i in range(process.appearance_time, time + process.full_completing_time)]
        process.timing[time:time + process.full_completing_time] = \
            ['И' for i in range(time, time + process.full_completing_time)]
        process.latency = process.timing.count('Г')
        time += process.full_completing_time
        process.full_completing_time = 0
        return process, time
```

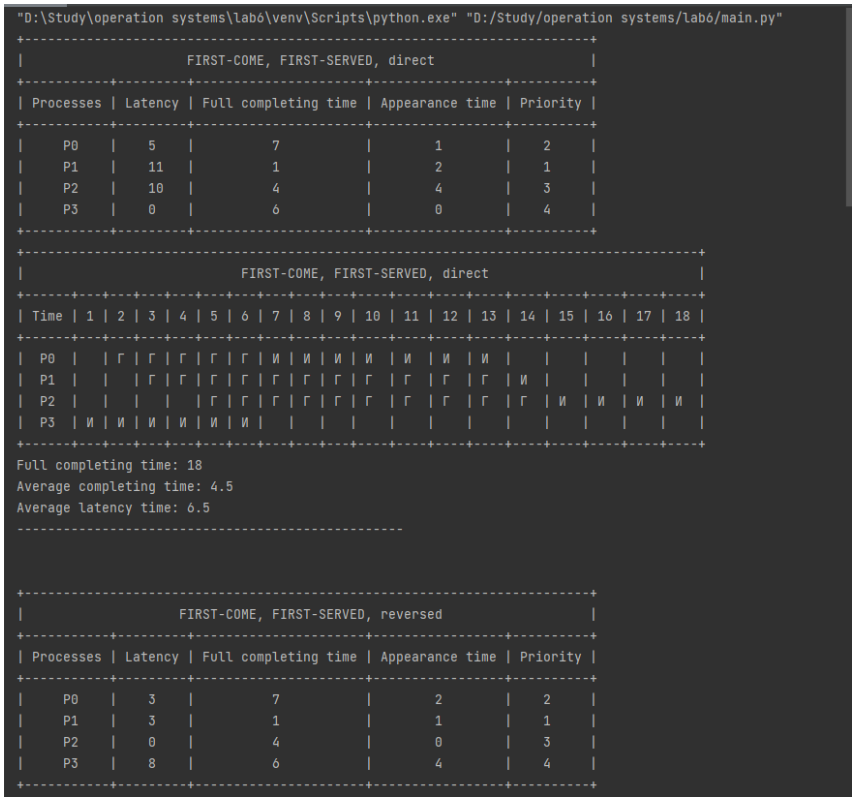
ShortestJobFirstPriority:

```
import copy
from Entities.BasePlanningEntity import BasePlanningEntity


class ShortestJobFirstPriority(BasePlanningEntity):
    __title__ = "Shortest Job First Priority"

    def __init__(self, processes: list):
        super().__init__(processes)

    def __make_plan__(self) -> None:
        super().__make_plan__()
        processes_copy = copy.deepcopy(self.__processes__)
        time = 0
        while time < self.__whole_time__:
            active_processes = [
                process for process in processes_copy
                if
                process.full_completing_time != 0 and process.appearance_time <= time
            ]
            active_processes.sort(key=lambda x: x.priority)
            if len(active_processes) > 0:
                active_processes[0] = self.__lil_beast__(active_processes[0], time)
            time += 1
        self.__point_ready_moments__(processes_copy)
        for i, process in enumerate(processes_copy, start=0):
            self.__processes__[i].latency = process.latency
            self.__processes__[i].timing = process.timing

    def __lil_beast__(self, process, *args):
        """
        Lil beast itself.
        :param process: process
        :param args: time
        :return: updated process
        """
        time = args[0]
        process.timing[time:time + 1] = ['И' for i in range(time, time + 1)]
        process.full_completing_time -= 1
        return process

    @staticmethod
    def __point_ready_moments__(processes):
        """
        It should be used after __lil_beast__.
        :param processes: processes
        :return: modified processes
        """
        for i, process in enumerate(processes, start=0):
            start = process.appearance_time
            end = ''.join(process.timing).rfind('И')
            for j in range(start, end + 1):
                if process.timing[j] != 'И':
                    process.timing[j] = 'Г'
                    process.latency += 1
        return processes
```

main:

```
from PlanningSystems.FCFSDirect import FCFSDirect
from PlanningSystems.FCFSReversed import FCFSReversed
from PlanningSystems.RoundRobin import RoundRobin
from PlanningSystems.ShortestJobFirstCrowdOut import ShortestJobFirstCrowdOut
from PlanningSystems.ShortestJobFirstNonCrowdOutPriority import ShortestJobFirstNonCrowdOutPriority
from PlanningSystems.ShortestJobFirstNonCrowdOut import ShortestJobFirstNonCrowdOut
from PlanningSystems.ShortestJobFirstPriority import ShortestJobFirstPriority
from Entities.ProcessEntity import ProcessEntity
import copy


def main():
    processes = [
        ProcessEntity("P0", 0, 7, 2, 1),
        ProcessEntity("P1", 0, 1, 1, 2),
        ProcessEntity("P2", 0, 4, 3, 4),
        ProcessEntity("P3", 0, 6, 4, 0),
    ]
    FCFSDirect(copy.deepcopy(processes)).show_info()
    FCFSReversed(copy.deepcopy(processes)).show_info()
    RoundRobin(copy.deepcopy(processes)).show_info()
    ShortestJobFirstNonCrowdOut(copy.deepcopy(processes)).show_info()
    ShortestJobFirstNonCrowdOutPriority(copy.deepcopy(processes)).show_info()
    ShortestJobFirstCrowdOut(copy.deepcopy(processes)).show_info()
    ShortestJobFirstPriority(copy.deepcopy(processes)).show_info()


if __name__ == "__main__":
    main()
```

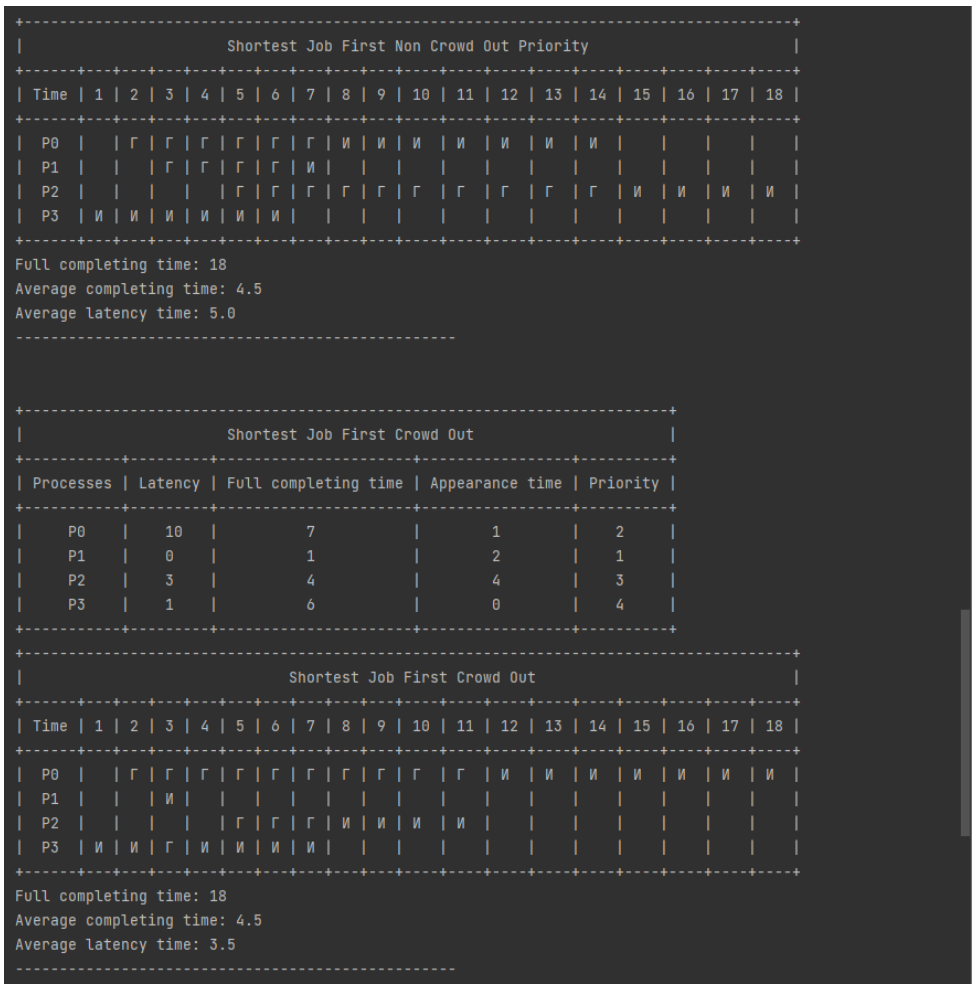
Результат работы программы:



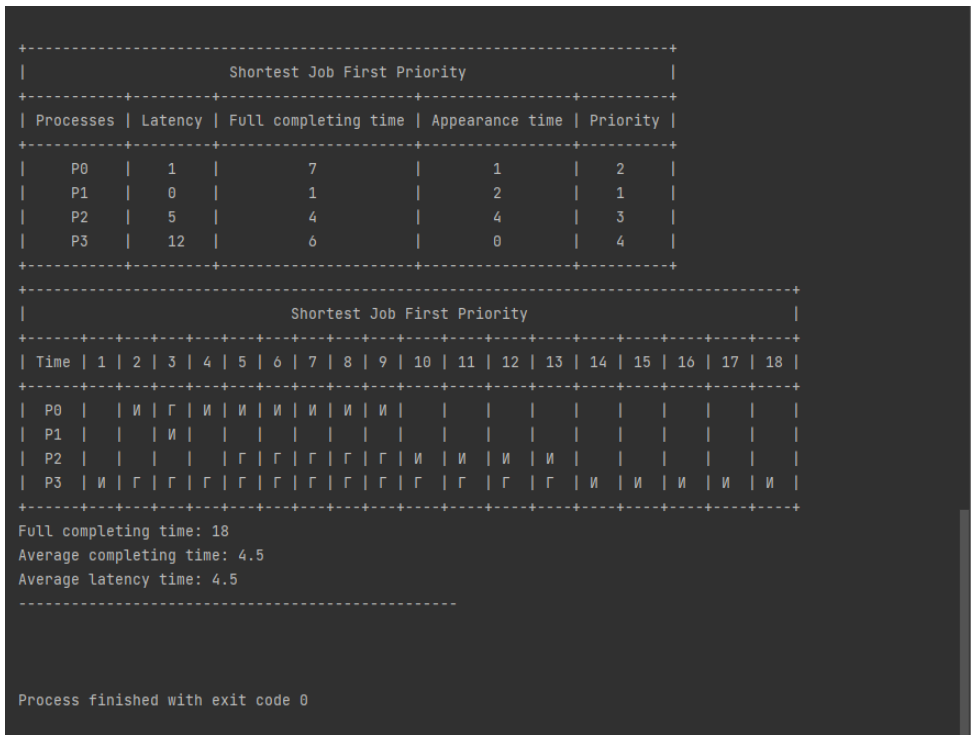
```
"D:\Study\operation systems\labó\venv\Scripts\python.exe" "D:/Study/operation systems/labó/main.py"
+-------------------------------------------------------------------------+
|                    FIRST-COME, FIRST-SERVED, direct                     |
+-----------+---------+----------------------+-----------------+----------+
| Processes | Latency | Full completing time | Appearance time | Priority |
+-----------+---------+----------------------+-----------------+----------+
|     P0    |    5    |          7           |        1        |    2     |
|     P1    |    11   |          1           |        2        |    1     |
|     P2    |    10   |          4           |        4        |    3     |
|     P3    |    0    |          6           |        0        |    4     |
+-----------+---------+----------------------+-----------------+----------+
+-----------------------------------------------------------------------------------------+
|                           FIRST-COME, FIRST-SERVED, direct                              |
+------+---+---+---+---+---+---+---+---+---+----+----+----+----+----+----+----+----+----+
| Time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
+------+---+---+---+---+---+---+---+---+---+----+----+----+----+----+----+----+----+----+
|  P0  |   | Г | Г | Г | Г | Г | И | И | И | И  | И  | И  | И  |    |    |    |    |    |
|  P1  |   |   | Г | Г | Г | Г | Г | Г | Г | Г  | Г  | Г  | Г  | И  |    |    |    |    |
|  P2  |   |   |   |   | Г | Г | Г | Г | Г | Г  | Г  | Г  | Г  | Г  | И  | И  | И  | И  |
|  P3  | И | И | И | И | И | И |   |   |   |    |    |    |    |    |    |    |    |    |
+------+---+---+---+---+---+---+---+---+---+----+----+----+----+----+----+----+----+----+
Full completing time: 18
Average completing time: 4.5
Average latency time: 6.5
---------------------------------------------------



+-------------------------------------------------------------------------+
|                   FIRST-COME, FIRST-SERVED, reversed                    |
+-----------+---------+----------------------+-----------------+----------+
| Processes | Latency | Full completing time | Appearance time | Priority |
+-----------+---------+----------------------+-----------------+----------+
|     P0    |    3    |          7           |        2        |    2     |
|     P1    |    3    |          1           |        1        |    1     |
|     P2    |    0    |          4           |        0        |    3     |
|     P3    |    8    |          6           |        4        |    4     |
+-----------+---------+----------------------+-----------------+----------+
```

```
+-----------------------------------------------------------------------------------+
|                          FIRST-COME, FIRST-SERVED, reversed                       |
+------+---+---+---+---+---+---+---+---+---+----+----+----+----+----+----+----+----+
| Time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
+------+---+---+---+---+---+---+---+---+---+----+----+----+----+----+----+----+----+----+
|  P0  |   |   | Г | Г | Г | И | И | И | И | И  | И  | И  |    |    |    |    |    |    |
|  P1  |   | Г | Г | Г | И |   |   |   |   |    |    |    |    |    |    |    |    |    |
|  P2  | И | И | И | И |   |   |   |   |   |    |    |    |    |    |    |    |    |    |
|  P3  |   |   |   |   | Г | Г | Г | Г | Г | Г  | Г  | Г  | И  | И  | И  | И  | И  | И  |
+------+---+---+---+---+---+---+---+---+---+----+----+----+----+----+----+----+----+----+
Full completing time: 18
Average completing time: 4.5
Average latency time: 3.5
---------------------------------------------------


+-------------------------------------------------------------------+
|                          ROUND ROBIN (RR)                         |
+-----------+---------+----------------------+-----------------+----------+
| Processes | Latency | Full completing time | Appearance time | Priority |
+-----------+---------+----------------------+-----------------+----------+
|     P0    |    10   |          7           |        1        |    2     |
|     P1    |    4    |          1           |        2        |    1     |
|     P2    |    9    |          4           |        4        |    3     |
|     P3    |    7    |          6           |        0        |    4     |
+-----------+---------+----------------------+-----------------+----------+
+-----------------------------------------------------------------------------------+
|                                ROUND ROBIN (RR)                                   |
+------+---+---+---+---+---+---+---+---+---+----+----+----+----+----+----+----+----+
| Time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
+------+---+---+---+---+---+---+---+---+---+----+----+----+----+----+----+----+----+----+
|  P0  |   | Г | Г | И | И | И | Г | Г | Г | Г  | Г  | Г  | Г  | И  | И  | И  | Г  | И  |
|  P1  |   |   | Г | Г | Г | Г | И |   |   |    |    |    |    |    |    |    |    |    |
|  P2  |   |   |   |   | Г | Г | Г | И | И | И  | Г  | Г  | Г  | Г  | Г  | Г  | И  |    |
|  P3  | И | И | И | И | Г | Г | Г | Г | Г | Г  | И  | И  | И  |    |    |    |    |    |
+------+---+---+---+---+---+---+---+---+---+----+----+----+----+----+----+----+----+----+
Full completing time: 18
Average completing time: 4.5
Average latency time: 7.5
---------------------------------------------------
```

```
+-------------------------------------------------------------------+
|                   Shortest Job First Non Crowd Out                |
+-----------+---------+----------------------+-----------------+----------+
| Processes | Latency | Full completing time | Appearance time | Priority |
+-----------+---------+----------------------+-----------------+----------+
|     P0    |    10   |          7           |        1        |    2     |
|     P1    |    4    |          1           |        2        |    1     |
|     P2    |    3    |          4           |        4        |    3     |
|     P3    |    0    |          6           |        0        |    4     |
+-----------+---------+----------------------+-----------------+----------+
+-----------------------------------------------------------------------------------+
|                          Shortest Job First Non Crowd Out                         |
+------+---+---+---+---+---+---+---+---+---+----+----+----+----+----+----+----+----+
| Time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
+------+---+---+---+---+---+---+---+---+---+----+----+----+----+----+----+----+----+----+
|  P0  |   | Г | Г | Г | Г | Г | Г | Г | Г | Г  | Г  | И  | И  | И  | И  | И  | И  | И  |
|  P1  |   |   | Г | Г | Г | Г | И |   |   |    |    |    |    |    |    |    |    |    |
|  P2  |   |   |   |   | Г | Г | Г | И | И | И  | И  |    |    |    |    |    |    |    |
|  P3  | И | И | И | И | И | И |   |   |   |    |    |    |    |    |    |    |    |    |
+------+---+---+---+---+---+---+---+---+---+----+----+----+----+----+----+----+----+----+
Full completing time: 18
Average completing time: 4.5
Average latency time: 4.25
---------------------------------------------------


+-------------------------------------------------------------------+
|               Shortest Job First Non Crowd Out Priority           |
+-----------+---------+----------------------+-----------------+----------+
| Processes | Latency | Full completing time | Appearance time | Priority |
+-----------+---------+----------------------+-----------------+----------+
|     P0    |    6    |          7           |        1        |    2     |
|     P1    |    4    |          1           |        2        |    1     |
|     P2    |    10   |          4           |        4        |    3     |
|     P3    |    0    |          6           |        0        |    4     |
+-----------+---------+----------------------+-----------------+----------+
```



```
                    Shortest Job First Non Crowd Out Priority
| Time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|  P0  |   | Г | Г | Г | Г | Г | Г | И | И | И  | И  | И  | И  | И  |    |    |    |    |
|  P1  |   |   | Г | Г | Г | Г | И |   |   |    |    |    |    |    |    |    |    |    |
|  P2  |   |   |   |   | Г | Г | Г | Г | Г | Г  | Г  | Г  | Г  | Г  | И  | И  | И  | И  |
|  P3  | И | И | И | И | И | И |   |   |   |    |    |    |    |    |    |    |    |    |
Full completing time: 18
Average completing time: 4.5
Average latency time: 5.0
-------------------------------------------------


                    Shortest Job First Crowd Out
| Processes | Latency | Full completing time | Appearance time | Priority |
|    P0     |    10   |          7           |        1        |    2     |
|    P1     |    0    |          1           |        2        |    1     |
|    P2     |    3    |          4           |        4        |    3     |
|    P3     |    1    |          6           |        0        |    4     |
                          Shortest Job First Crowd Out
| Time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|  P0  |   | Г | Г | Г | Г | Г | Г | Г | Г | Г  | Г  | И  | И  | И  | И  | И  | И  | И  |
|  P1  |   |   | И |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |
|  P2  |   |   |   |   | Г | Г | Г | И | И | И  | И  |    |    |    |    |    |    |    |
|  P3  | И | И | Г | И | И | И | И |   |   |    |    |    |    |    |    |    |    |    |
Full completing time: 18
Average completing time: 4.5
Average latency time: 3.5
-------------------------------------------------
```



```
                    Shortest Job First Priority
| Processes | Latency | Full completing time | Appearance time | Priority |
|    P0     |    1    |          7           |        1        |    2     |
|    P1     |    0    |          1           |        2        |    1     |
|    P2     |    5    |          4           |        4        |    3     |
|    P3     |    12   |          6           |        0        |    4     |
                          Shortest Job First Priority
| Time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|  P0  |   | И | Г | И | И | И | И | И | И |    |    |    |    |    |    |    |    |    |
|  P1  |   |   | И |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |
|  P2  |   |   |   |   | Г | Г | Г | Г | Г | И  | И  | И  | И  |    |    |    |    |    |
|  P3  | И | Г | Г | Г | Г | Г | Г | Г | Г | Г  | Г  | Г  | Г  | И  | И  | И  | И  | И  |
Full completing time: 18
Average completing time: 4.5
Average latency time: 4.5
-------------------------------------------------



Process finished with exit code 0
```

Вывод: в результате выполнения лабораторной работы был изучены типовые алгоритмы планирования.