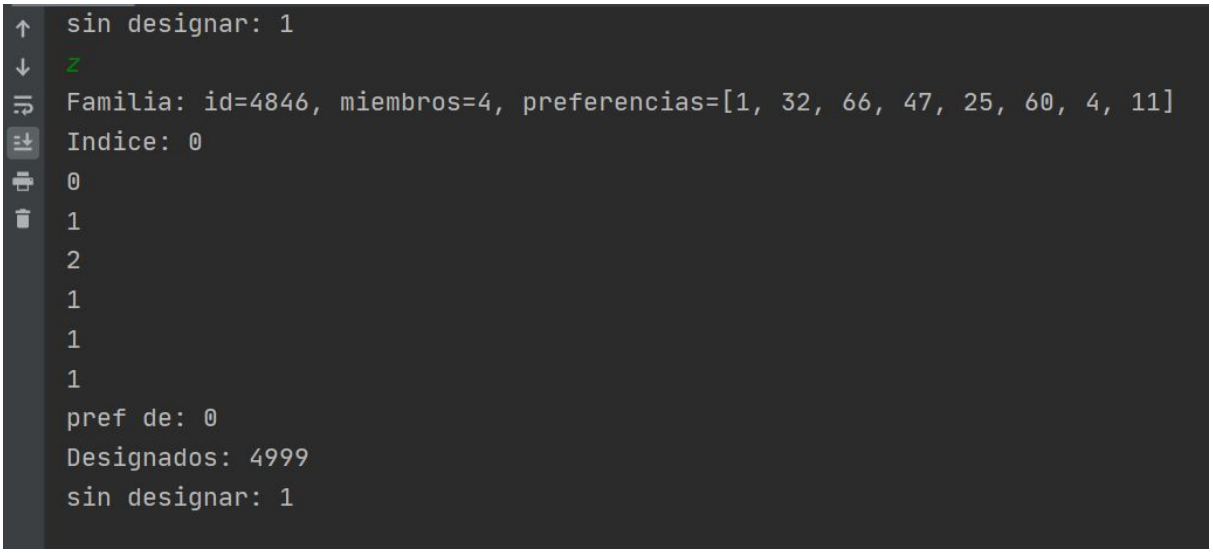
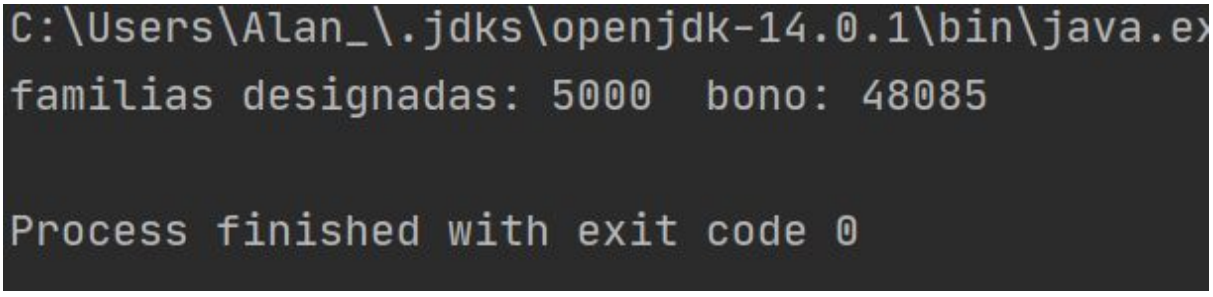
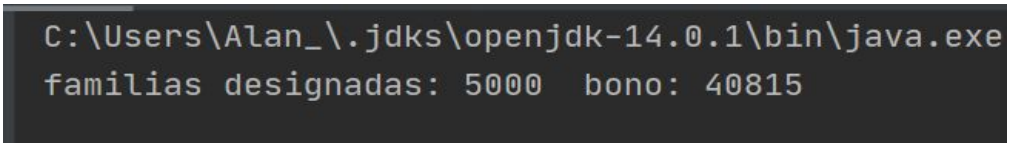
Comencé a pensar el código de la siguiente manera, la primerísima versión de pensamiento fue “Recorro la lista de familias con el iterador, designó a cada una en su día preferido, los que no pudieron ser designados lo guardo en un arreglo auxiliar y les seteo el índice del preferido en +1 y recorro hasta que se designen todos”
Utilizo una clase Dia la cual va guardando el total de familias que posee, añade una familia y le resta a su capacidad actual los miembros de la misma.
También un arreglo estático Días[] arDias para usar accesos estáticos de memoria
Luego de varias pruebas y modificaciones siempre llegue a que me faltaba 1 para designar pero sin espacio en los días de preferencia a lo que ordene el arreglo de familias de mayor a menor pero sin solución aparente



Lista que aparece después de índice: 0 son los espacios disponibles en sus días de preferencia con lo que decidí re encarar el código, este mismo queda guardado en una banch llamada “versiónPrimera” (La cual está sucia de sistems output y demás)
Comienzo el refactor pero está vez un poco más organizado, habiendo modificado algunas cosas en familia y en la clase Días pude lograr un resultado. Primero pensé en ordenar el arreglo de familias ocn collection.sort por la cantidad de integrantes de la familia (ya que pensando en bueno agrego a todas las familias con más cantidad de integrantes en sus días preferidos reduciendo el bono) dando este resultado:



Gracias a un contador dentro de días que cada vez que se le asigna una familia, recorriendo la colección de días solicitando la cantidad de familias de c/u arroja ese resultado dando como bono 48085. Solo por curiosidad me pregunte que pasaria si no ordenaba el arreglo(lo cual le agrega también complejidad computacional) resultando en una reducción del bono.



Entre algunos intentos por reducir el bono llegue a qué reduciendo también la cantidad de días de cada familia podía lograr un buen resultado, usar 6(de sus 8) días era poco porque siempre faltaban designar 3 familias, para resolver esto debía “hardcodear” lo cual no me parecio buena idea

```
        int mejorDia = 0;
        int capacidad = 0;
        for (int i = 0; i < 6; i++) {
            if (capacidad < dias[a.diaPreferido()-1]
Main ×
C:\Users\Alan_\.jdks\openjdk-14.0.1\bin\java.exe "-ja
familias designadas: 4997  bono: 39135
```

Lo que hace el algoritmo en este punto es si no te designaron tu dia preferido busca en tu arreglo de dias el dia que tenga más capacidad disponible

```
        int mejorDia = 0;
        int capacidad = 0;
        for (int i = 0; i < 8; i++) {
            if (capacidad < dias[a.diaPreferi
Main ×
C:\Users\Alan_\.jdks\openjdk-14.0.1\bin\java.e
familias designadas: 5000  bono: 42375
```

Utilizando los 8 días aumenta considerablemente el bono a entregar, se designan todas las familias

```
        int mejorDia = 0;
        int capacidad = 0;
        for (int i = 0; i < 7; i++) {
            if (capacidad < dias[a.diaPre
Main ×
C:\Users\Alan_\.jdks\openjdk-14.0.1\bin\j
familias designadas: 5000  bono: 40815

Process finished with exit code 0
```

Pero al usar solo 7(de 8 días) logre designar a todas las familias reduciendo el bono compensatorio a 40.815
algo lejos del resultado de la arrojado por la cátedra

Con esto queda finalizada la “versionSegunda” como nombre de la rama en la que se va a guardar y comienzo un refactor para lograr minimizar algo más el resultado del bono

Por último y gracias a los intercambios entre compañeros llegue a que es mejor asignar primero a todas las pequeñas familias y luego a todas las demás preguntando sí en su primer dia de preferencia hay lugar, sino el siguiente y asi dando este resultado(final):

```
familias designadas: 5000  bono: 36660

Process finished with exit code 0
```