Universidad de Mendoza
Facultad de Ingeniería
Asignatura: Programación II
Carreras: Ing. Informática
2018

**Guía de Trabajos Prácticos 2018**

**Trabajo Práctico No 1**

Consigna:

1. Registrar un usuario en https://github.com/
2. Generar un repositorio remoto Programacion 2.Trabajo Practico 01.Consigna 01
3. Crear una clase Persona con los atributos
    a. long dni;
    b. string apellido;
    c. string nombre;
4. Escribir los métodos Getter/Setter de todos los atributos
5. Agregar el método input para llenar los atributos del objeto
6. Agregar el método show para mostrar los atributos del objeto
7. Redefinir método toString de la clase Persona
8. Hacer un git push de todas las modificaciones del repositorio local en el repositorio remoto

9. Generar un repositorio remoto Programacion 2.Trabajo Practico 01.Consigna 02
10. Crear una clase Fecha con 3 atributos
    a. int dia;
    b. int mes;
    c. int anho;
11. Escribir los métodos Getter/Setter de todos los atributos
12. Agregar el método input para llenar los atributos del objeto
13. Agregar el método show para mostrar los atributos del objeto
14. Redefinir método toString de la clase Fecha
15. Escribir el método comparar que me permita comparar 2 Fecha devolviendo -1 si fecha1 < fecha2, 0 si las fechas son iguales y 1 si fecha1 > fecha2
16. Escribir un método desplazar que permita sumar o restar a una Fecha un número entero que corresponde a la cantidad de días que se quiere desplazar
17. Hacer un git push de todas las modificaciones del repositorio local en el repositorio remoto

18. Generar un repositorio remoto Programacion 2.Trabajo Practico 01.Consigna 03
19. Crear una clase PersonaConNacimiento que extienda de la clase Persona con el atributo adicional
    a. Fecha nacimiento;
20. Escribir los métodos Getter/Setter de todos los atributos
21. Agregar el método input para llenar los atributos del objeto
22. Agregar el método show para mostrar los atributos del objeto
23. Redefinir método toString de la clase PersonaConNacimiento
24. Hacer un git push de todas las modificaciones del repositorio local en el repositorio remoto

**Guía de Trabajos Prácticos 2018**

**Trabajo Práctico No 2**

Consigna:

1. Generar un repositorio remoto Programacion 2.Trabajo Practico 02.Consigna 01
2. Crear una clase Materia con los atributos
    a. private String codigoMateria;
    b. private String nombre;
3. Escribir los métodos Getter/Setter de todos los atributos
4. Sobrecargar el constructor para crear los objetos con valores predeterminados
5. Redefinir el método toString de la clase Materia
6. Escribir las pruebas de test para los métodos

7. Crear una clase Cargo (titular, adjunto, jtp) con los atributos
    a. private String nombre;
    b. private double basico;
8. Escribir los métodos Getter/Setter de todos los atributos
9. Sobrecargar el constructor para crear los objetos con valores predeterminados
10. Redefinir método toString de la clase Cargo
11. Escribir las pruebas de test para los métodos

12. Crear una clase CargoMateria con los atributos
    a. private Materia materia;
    b. private Cargo cargo;
13. Escribir los métodos Getter/Setter de todos los atributos
14. Sobrecargar el constructor para inyectarle las dependencias de los objetos con valores predeterminados
15. Redefinir método toString de la clase Cargo
16. Escribir las pruebas de test para los métodos

17. Crear una clase Profesor con los atributos
    a. private int profesor_id;
    b. private String nombre;
    c. private String apellido;
    d. private CargoMateria[] cargomateria;
18. Escribir los métodos Getter/Setter de todos los atributos
19. Sobrecargar el constructor para asignarle valores predeterminados
20. Redefinir método toString de la clase Profesor
21. Escribir las pruebas de test para los métodos

22. Crear una clase ContenidoTeorico con el atributo

**Guía de Trabajos Prácticos 2018**

   a. private String tema;
   b. private String descripcion;
   c. private int horasPresenciales;
23. Escribir los métodos Getter/Setter de todos los atributos
24. Sobrecargar el constructor para asignarle valores predeterminados
25. Redefinir método toString de la clase ContenidoTeorico
26. Escribir las pruebas de test para los métodos

27. Crear una clase TrabajoPractico con el atributo
   a. private int numero;
   b. private String ejercitacion;
   c. private int horasLaboratorio;
28. Escribir los métodos Getter/Setter de todos los atributos
29. Sobrecargar el constructor para asignarle valores predeterminados
30. Redefinir método toString de la clase TrabajoPractico
31. Escribir las pruebas de test para los métodos

32. Extender la clase Profesor creando la clase ProfesorTitular agregando el atributo
   a. private ContenidoTeorico contenidoTeorico;
33. Escribir los métodos Getter/Setter de todos los atributos
34. Sobrecargar el constructor para inyectarle las dependencias de los objetos con valores predeterminados
35. Redefinir método toString de la clase ProfesorTitular
36. Escribir las pruebas de test para los métodos

37. Extender la clase Profesor creando la clase ProfesorAdjunto agregando el atributo
   a. private TrabajoPractico trabajoPractico;
38. Escribir los métodos Getter/Setter de todos los atributos
39. Sobrecargar el constructor para inyectarle las dependencias de los objetos con valores predeterminados
40. Redefinir método toString de la clase ProfesorAdjunto
41. Escribir las pruebas de test para los métodos

42. Crear la clase ComunidadDocente con el atributo
   a. private Profesor[] profesores;
43. Agregar un método add() que permita agregar Profesor al arreglo sin dejar huecos
44. Agregar un método remove() que permita eliminar Profesor del arreglo sin dejar huecos
45. Agregar un método show() que muestre todos los profesores demostrando el polimorfismo
46. Escribir las pruebas de test para los métodos

**Guía de Trabajos Prácticos 2018**

47. Crear la clase ProfesorService
48. Agregar un método show(Profesor profesor) que muestre los datos de un Profesor demostrando el polimorfismo
49. Escribir las pruebas de test para los métodos

50. Crear la clase Curso con los atributos
    a. private String ubicacion;
    b. private int capacidad;
51. Escribir los métodos Getter/Setter de todos los atributos
52. Sobrecargar el constructor para asignar valores predeterminados
53. Redefinir método toString de la clase Curso
54. Escribir las pruebas de test para los métodos

55. Definir la interface Cursable con el método
    a. public void asignaCurso(Curso curso);

56. Crear la clase Alumno con los siguientes atributos
    a. private int legajo;
    b. private String nombre;
    c. private String apellido;
57. Escribir los métodos Getter/Setter de todos los atributos
58. Sobrecargar el constructor para asignar valores predeterminados
59. Redefinir método toString de la clase Alumno
60. Escribir las pruebas de test para los métodos

61. Las clases Alumno, Profesor, ProfesorTitular y ProfesorAdjunto deben implementar la interface Cursable;

62. Crear una clase ComunidadEducativa que tenga el atributo
    a. private Cursable[] comunidad;
63. Agregar un método add() que permita agregar cualquier tipo asociado sin dejar huecos
64. Agregar un método remove() que permita eliminar cualquier tipo asociado sin dejar huecos
65. Agregar un método show() que permita listar todos los objetos contenidos
66. Escribir las pruebas de test para los métodos

67. Hacer un git push de todas las modificaciones del repositorio local en el repositorio remoto

**Guía de Trabajos Prácticos 2018**

**Consigna:**
Desarrollar nuestras propias implementaciones de colecciones; en este caso un equivalente a List.
Las clases utilizarán generics para manipular los objetos.
Debemos crear una interfaz y dos implementaciones de la interfaz.

La interfaz deberá tener los siguientes métodos:

1. void agregar(T elemento)
2. T obtener(int posicion)
3. void borrar(int posicion)
4. void borrarTodos()

**Implementación 1**
* Almacenará los elementos en un arreglo, que inicialmente es de tamaño 5.
* Se comenzará a guardar datos en la posición 0 y se irá incrementando a medida que se guarden nuevos datos.
* Si el arreglo tiene los 5 lugares ocupados y se quiere agregar un nuevo elemento se incrementará en 5 el arreglo. Esto se tendrá que hacer creando un nuevo arreglo y copiando los elementos del arreglo chico al grande.
* Si se elimina un elemento del arreglo se deberá compactar. Ejemplo:
Supongamos un arreglo de letras [a,b,c,d,vacío], si eliminamos la letra b el arreglo debería quedar compactado de la siguiente forma [a,c,d,vacío,vacío].

**Implementación 2**
* Almacenará los datos en una lista simplemente enlazada.
* Se utilizará un objeto contenedor donde se almacena el objeto con el dato y un puntero al próximo contenedor en la lista.
* Los elementos se agregarán de manera que la lista quede siempre ordenada.

**Nota:**
Ambas implementaciones se probarán con al menos 2 clases propias.