Objectius:
- Aprendre a serialitzar objectes JAVA

Instruccions:
- Responeu a l'espai de cada pregunta, si ho feu amb diapositives o captures d'images enganxeu la diapositiva en aquest mateix espai.
- Es valorarà la presentació i els comentaris al codi

Criteris d'avaluació:
- Cada exercici té la mateixa puntuació
- Les metodologies de treball pròpies, organització personal i participació valen un 10%

Entrega:
- Un arxiu .zip anomenat: **PRx.y-NomCognom.zip**
  - PRx.y correspon al codi de la pràctica, per exemple PR1.1
  - NomCognom correspon al nom i primer cognom de cada participant
- L'arxiu **.zip** conte:
  - Aquest document emplenat en format **.pdf** anomenat **memoria.pdf**
  - Els arxius necessaris per fer anar la pràctica
- Esteu indicant l'enllaç al repositori Git

Nom i Cognom: Patricio André Rojas Condori
Enllaç al vostre repositori Git:
https://github.com/PatricioGitHub1/PR1-2-SerialitzacioObjectes

Materials:
Necessiteu una eina per programar en JAVA
Feu servir Google per buscar els tutorials que us serveixin millor
El repositori bàsic és l'usat també en la pràctica anterior
https://github.com/optimisme/DAM-JavaPersitenciaFitxers

Tasques, a cada exercici feu l'explicació i captures que cregueu convenients

- Preparació - Continueu afegint codi en el menú de java que vau preparar per la pràctica anterior.:

```java
import java.io.IOException;
import java.util.*;

public class Main {
  static Scanner in = new Scanner(System.in); // System.in és global

  // Main
  public static void main(String[] args) throws InterruptedException, IOException {
    boolean running = true;
    while (running) {
      String menu = "Escull una opció:";
      menu = menu + "\n 0) PR120ReadFile";
      menu = menu + "\n 1) PR121Files";
      // Adapta aquí les altres classes de l'exercici (PR122cat…)
      menu = menu + "\n 100) Sortir";
      System.out.println(menu);

      int opcio = Integer.valueOf(llegirLinia("Opció:"));
      try {
        switch (opcio) {
          case 0: PR120ReadFile.main(args); break;
          case 1: PR121Files.main(args); break;
          // Adapta aquí les altres classes de l'exercici (PR122cat…)
          case 100: running = false; break;
          default: break;
        }
      } catch (Exception e) {
          System.out.println(e);
      }
    }
    in.close();
  }

  static public String llegirLinia (String text) {
    System.out.print(text);
    return in.nextLine();
  }
}
```

- Exercici 0

PR130mainPersonesHashmap.java

- Crea un HashMap<String, Integer> amb el nom i l'edat de 5 persones (dades predefinides).
- Empra DataOutputStream per guardar aquestes dades en un arxiu **PR130persones.dat**.
- Llegeix **PR130persones.dat** amb DataInputStream i mostra el seu contingut per pantalla.

(Mirar exemple EscripturaDadesPrimitives.java i LecturaDadesPrimitives.java)

Primer declarem l'objecte amb el qual guardarem el HashMap i posteriorment hi afegim diferentes parelles clau - valor.

```java
public class PR130mainPersonesHashmap {
    HashMap<String, Integer> map = new HashMap<String, Integer>();

    Run | Debug
    public static void main(String[] args) {
        PR130mainPersonesHashmap people = new PR130mainPersonesHashmap();
        people.map.put("Jhon", 19);
        people.map.put("Lia", 3);
        people.map.put("Dana", 2);
        people.map.put("Christian", 45);
        people.map.put("Andrew", 29);

        people.escripturaDadesPrimitives();
        people.lecturaDadesPrimitives();
    }

    void escripturaDadesPrimitives() {
        String basePath = System.getProperty("user.dir") + "/data/";
        String filePath = basePath + "PR130persones.dat";

        // Crear la carpeta 'data' si no existeix
        File dir = new File(basePath);
        if (!dir.exists()){
            if(!dir.mkdirs()) {
                System.out.println("Error en la creació de la carpeta 'data'");
            }
        } else {
            System.out.println("La carpeta 'Data' existeix");
        }

        // Escriptura en l'arxiu 'PR130persones.dat'
        try {
            FileOutputStream fos = new FileOutputStream(filePath);
            DataOutputStream dos = new DataOutputStream(fos);

            // Iterar sobre el HashMap per treure les dades
            for (Iterator i = this.map.keySet().iterator(); i.hasNext();) {
                String name = (String) i.next();
```

El següent són les dues funciones, la primera serveix per escriure aquestes dades primitives en l'arxiu .dat
La segona llegeix aquest últim arxiu i ens permet interactuar amb el seu contingut i mostrar-lo per consola.

```
            FileOutputStream fos = new FileOutputStream(filePath);
            DataOutputStream dos = new DataOutputStream(fos);

            // Iterar sobre el HashMap per treure les dades
            for (Iterator i = this.map.keySet().iterator(); i.hasNext();) {
                String name = (String) i.next();
                int age = (int) this.map.get(name);

                dos.writeUTF(name);
                dos.writeInt(age);
                System.out.println("Entrada añadida");
            }

            // Importante, al acabar usar el flush() para subir la informacion y cerrar
            dos.flush();
            fos.close();
            dos.close();
        }
        catch (FileNotFoundException e1) {
            e1.printStackTrace();
        }
         catch ( IOException e2) {
            e2.printStackTrace();
        }
    }

    void lecturaDadesPrimitives() {
        // Conseguim direccions arxiu
        String basePath = System.getProperty("user.dir") + "/data/";
        String filePath = basePath + "PR130persones.dat";

        try {
            // Declarem les instancies necessaries
            FileInputStream fis = new FileInputStream(filePath);
            DataInputStream dis = new DataInputStream(fis);

            // Iterem en funcio del HashMap que hem creat i mostrem les dades
            for (int i = 0; i != this.map.size(); i++) {
                System.out.println("Nombre = "+dis.readUTF()+" | Edat = "+dis.readInt())
            }

            // Tanquem les funciones
            dis.close();
            fis.close();
        }   catch (FileNotFoundException e2) {
            e2.printStackTrace();
        } catch (IOException e1) {
            e1.printStackTrace();
```

- Exercici 1

Crea una classe **PR131hashmap** que implementa Serializable i conté un HashMap. Crea dos procediments:

- **PR131mainEscriu.java**: Escriu el HashMap a PR131HashMapData.ser.
- **PR131mainLlegeix.java**: Llegeix PR131HashMapData.ser i mostra el seu contingut per pantalla.

(Mirar exemples EscripturaObjectes.java i LecutraObjectes.java)

Aqui esta la clase que implementa Serializable

```java
import java.io.Serializable;
import java.util.HashMap;

public class PR131hashmap implements Serializable {
    HashMap<String, String> hsmp = new HashMap<String, String>();

    PR131hashmap(String key, String value) {
        for (int i = 1; i != 7; i++) {
            hsmp.put(key+"_"+i, value+"_"+i);
        }
    }
}
```

Aquest seria el la classe per escriure aquest HashMap en el document .dat

```java
public class PR131mainEscriu {
    Run | Debug
    public static void main(String[] args) {
        String basePath = System.getProperty("user.dir") + "/data/";
        String filePath = basePath + "PR131HashMapData.ser";

        // Crear la carpeta 'data' si no existeix
        File dir = new File(basePath);
        if (!dir.exists()){
            if(!dir.mkdirs()) {
                System.out.println("Error en la creació de la carpeta 'data'");
            }
        }

        System.out.println("");

        try {
            FileOutputStream fos = new FileOutputStream(filePath);
            ObjectOutputStream oos = new ObjectOutputStream(fos);

            PR131hashmap obj0 = new PR131hashmap(key:"llave", value:"valor");

            oos.writeObject(obj0);

            oos.close();
            fos.close();

            System.out.println("Objeto escrito");

        } catch (IOException e) { e.printStackTrace(); }
    }
}
```

I finalment el document per llegit aquest fitxer i extreure l'objecte HashMap.

```java
public class PR131mainLlegeix {
    Run | Debug
    public static void main(String[] args) {
        String basePath = System.getProperty("user.dir") + "/data/";
        String filePath = basePath + "PR131HashMapData.ser";

        // Crear la carpeta 'data' si no existeix
        File dir = new File(basePath);
        if (!dir.exists()){
            if(!dir.mkdirs()) {
                System.out.println("Error en la creació de la carpeta 'data'");
            }
        }

        System.out.println("");

        try {
            FileInputStream fis = new FileInputStream(filePath);
            ObjectInputStream ois = new ObjectInputStream(fis);

            PR131hashmap obj0 = (PR131hashmap) ois.readObject();

            System.out.println("Datos del HashMap\n================");
            HashMap<String, String> hm = obj0.hsmp;

            for (Map.Entry<String, String> set :
             hm.entrySet()) {
                System.out.println(set.getKey() + " = " + set.getValue());
            }

            ois.close();
            fis.close();

        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) { e.printStackTrace(); }
    }
}
```

- Exercici 2

Crea una clase **'PR132persona'** que implementi Serializable amb els atributs: Nom, Cognom, Edat. Després fes un programa "**PR132main.java**" amb objectes que tinguin les següents dades (que es mostren a la taula) i guarda'ls en un arxiu "**PR132people.dat**". Finalment llegeix l'arxiu que s'acaba de guardar i mostra la informació per pantalla:

Nom	**Cognom**	**Edat**
Maria	López	36
Gustavo	Ponts	63
Irene	Sales	54

(Mirar exemples EscripturaObjectes.java i LecutraObjectes.java,
igual que l'exercici anterior)

Primer creem la clase que implementa Serializable.

```java
public class PR132persona implements Serializable{
    String nom;
    String cognom;
    int edat;

    public PR132persona(String nom, String cognom, int edat) {
        this.nom = nom;
        this.cognom = cognom;
        this.edat = edat;
    }

    @Override
    public String toString() {
        return "Nom = " + nom + " | Cognom = " + cognom + " | Edat = " + edat;
    }
}
```

Codi de la classe on guardem els objectes en el document .dat i posteriorment els llegim i mostrem per consola

```java
public class PR132main {
    Run | Debug
    public static void main(String[] args) {
        String basePath = System.getProperty("user.dir") + "/data/";
        String filePath = basePath + "PR132people.dat";

        // Crear la carpeta 'data' si no existeix
        File dir = new File(basePath);
        if (!dir.exists()){
            if(!dir.mkdirs()) {
                System.out.println("Error en la creació de la carpeta 'data'");
            }
        }

        System.out.println("");

        try {
            // Escribir los objetos
            FileOutputStream fos = new FileOutputStream(filePath);
            ObjectOutputStream oos = new ObjectOutputStream(fos);


            oos.writeObject(new PR132persona(nom:"Maria", cognom:"López", edat:36));
            System.out.println("Objeto escrito");

            oos.writeObject(new PR132persona(nom:"Gustavo", cognom:"Ponts", edat:63))
            System.out.println("Objeto escrito");

            oos.writeObject(new PR132persona(nom:"Irene", cognom:"Sales", edat:54));
            System.out.println("Objeto escrito");

            oos.close();
            fos.close();

            System.out.println("Todos los objetos escritos en PR132people.dat\n");

            // Momento de leer los objetos
```

```
        try {
            System.out.println("Mostrando los objetos desde 132people.dat...");
            FileInputStream fis = new FileInputStream(filePath);
            ObjectInputStream ois = new ObjectInputStream(fis);

            PR132persona obj0 = (PR132persona) ois.readObject();
            PR132persona obj1 = (PR132persona) ois.readObject();
            PR132persona obj2 = (PR132persona) ois.readObject();

            System.out.println(obj0);
            System.out.println(obj1);
            System.out.println(obj2);

            ois.close();
            fis.close();

        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) { e.printStackTrace(); }

    } catch (IOException e) { e.printStackTrace(); }
}
```

- Exercici 3

Crea un programa "**PR133mainTreballadors.java**", i crea manualment un arxiu anomenat "**PR133treballadors.csv**", amb les dades de la taula següent.

Id	**Nom**	**Cognom**	**Departament**	**Salari**
123	Nicolás	Rana	2	1000.00
435	Xavi	Gil	2	1800.50
876	Daniel	Ramos	6	700.30
285	Pedro	Drake	4	2500.00
224	Joan	Potter	6	1000.00

Fes que el programa demani a l'usuari un identificador de treballador, quina dada vol modificar i el nou valor i faci la modificació al propi arxiu .csv

(Mirar exemple GestioCSV)

Fent servir funciones es va aconseguir fer aquesta tasca.

```java
public class PR133mainTreballadors {
    Run | Debug
    public static void main(String[] args) {
        String basePath = System.getProperty("user.dir") + "/data/";
        String fileName = "PR133treballadors.csv";
        String filePath = basePath + fileName;

        System.out.println("");

        // Array amb cada linia del document
        List<String> csv = UtilsCSV.read(filePath);

        // Scanner lee ID
        try {
            Scanner sc = new Scanner(System.in);
            System.out.print("ID del treballador: ");

            String id = sc.next();
            int numLiniaEmpleado = UtilsCSV.getLineNumber(csv, column:"Id", id);

            if (numLiniaEmpleado == -1) {
                throw new Exception("User not found");
            } else {
                sc.nextLine();
                // Sacar campos
                String[] columnes = UtilsCSV.getKeys(csv);

                System.out.print("Camp a canviar: ");
                String camp_canviar = sc.next();

                // Comprovar que existe el campo
                boolean campExists = false;
                for (String cmp: columnes) {
                    if (cmp.equals(camp_canviar)) {
                        campExists = true;
                        break;
                    }
                }

                if (!campExists) {
                    throw new Exception("Camp not found");
                }
                sc.nextLine();
                System.out.print("Nou valor: ");
                String new_camp = sc.next();
                // Hacer modificaciones
                UtilsCSV.update(csv, numLiniaEmpleado, camp canviar, new camp);
```

- Exercici 4 - Registre d'estudiants amb RandomAccessFile

Descripció:
Una universitat vol gestionar les notes dels seus estudiants de manera eficient. Cada estudiant té un número de registre únic (en format enter) i una nota final associada. Per

permetre un accés ràpid a les dades i poder actualitzar-les sense haver de carregar tot el fitxer a memòria o recórrer-lo completament, decideixen utilitzar RandomAccessFile.

Requisits:

1. El fitxer d'estudiants ha de tenir una estructura amb una longitud fixa per registre, permetent l'accés directe a les dades de qualsevol estudiant. Es suggerix una estructura on:
    - El número de registre ocupa 4 bytes (ja que és un enter).
    - El nom ocupa 20 caràcters.
    - La nota ocupa 4 bytes (en format float).
2. El programa ha de permetre a l'usuari:
    - Afegir un nou estudiant amb la seva nota.
    - Actualitzar la nota d'un estudiant existent mitjançant el seu número de registre.
    - Consultar la nota d'un estudiant mitjançant el seu número de registre.
3. L'accés i modificació de les dades han de ser eficients, evitant recórrer tot el fitxer si no és necessari.

Consideracions addicionals:

- Cal gestionar possibles errors, com ara intentar accedir a un estudiant que no existeix.
- El programa ha de garantir que les dades introduïdes estiguin en el format correcte.
- Encara que per aquest exercici pugui suposar-se un límit d'estudiants, el codi ha de ser adaptable per gestionar un nombre més gran d'entrades si es requereix.

Podeu consultar aquest exemple i fer-lo servir com a base:
https://docs.google.com/document/d/1AKQYnn9CeWwXEmvM_61_OaeM0BB_EbK2gBemO6S_RX8/edit?usp=sharing

```java
public class PR134estudiantsManager {
    private static final int ID_SIZE = 4; // bytes
    private static final int CHAR_SIZE = 2; // bytes per caràcter en UTF-16
    private static final int NAME_SIZE = 20; // Longitud màxima en caràcters del nom
    private static final int GRADE_SIZE = 4; //bytes

    Run | Debug
    public static void main(String[] args) {
        try (RandomAccessFile raf = new RandomAccessFile("./data/estudiants.dat", "rw")) {
            // Resetear el documento y poner los valores base
            restartDataFile(raf);

            // Afegim nou registre
            afegirEstudiant(raf, id:4, nom:"Antonio Perez", (float) 5.00);

            // Consultar i mostrar els estudiants afegits
            mostrarEstudiant(raf, id:1, msg:"Original");
            mostrarEstudiant(raf, id:2, msg:"Original");
            mostrarEstudiant(raf, id:3, msg:"Original");
            mostrarEstudiant(raf, id:4, msg:"Original");

            // Actualitzar els noms dels estudiants
            actualitzarNotaEstudiant(raf, id:1, (float) 5.50);
            actualitzarNotaEstudiant(raf, id:4, (float) 10.00);


            // Consultar i mostrar els estudiants actualitzats
            mostrarEstudiant(raf, id:1, msg:"Canvi a la nota");
            mostrarEstudiant(raf, id:4, msg:"Canvi a la nota");

            // Caso en donde el estudiante no existe
            mostrarEstudiant(raf, id:30, msg:"Estudiant no existeix");

        } catch (EOFException e){
            System.out.println("Aquest estudiant no existeix");
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```java
public static void restartDataFile(RandomAccessFile raf) {
    String basePath = System.getProperty("user.dir") + File.separator + "data" + File.separator + "estudiants.dat";
    File newFile = new File(basePath);
    // Crear o en su defecto vaciar el archivo de datos
    if (!newFile.exists()) {
        try {
            newFile.createNewFile();
            System.out.println("Se ha creado el archivo de datos");
        } catch (IOException e) {
            e.printStackTrace();
        }
    } else {
        FileWriter fileWriter;
        try {
            fileWriter = new FileWriter(basePath, false);
            fileWriter.close();
            System.out.println("Disponible archivo de datos");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    // Introducir los casos base
    try {
        afegirEstudiant(raf, id:1, nom:"Patricio Rojas", (float) 7.50);
        afegirEstudiant(raf, id:2, nom:"John Doe", (float) 4.99);
        afegirEstudiant(raf, id:3, nom:"Jane Doe", (float) 8.73);

    } catch (Exception e) {
        e.printStackTrace();
    }
}

public static void afegirEstudiant(RandomAccessFile raf, int id, String nom, float nota) throws Exception {
    raf.seek(raf.length());
    raf.writeInt(id);
    raf.writeChars(getPaddedName(nom));
    raf.writeFloat(nota);
}


public static String consultarEstudiante(RandomAccessFile raf, int id) throws Exception {
    raf.seek(getSeekPosition(id));
    raf.readInt();
    char[] chars = new char[NAME_SIZE];
    for (int i = 0; i < NAME_SIZE; i++) {
        chars[i] = raf.readChar();
    }
```

```
    return new String(chars).trim() + " - "+ Float.toString(raf.readFloat()) ;
}


public static void actualitzarNotaEstudiant(RandomAccessFile raf, int id, float novaNota) throws Exception {
    raf.seek(getSeekPosition(id) + ID_SIZE + NAME_SIZE * CHAR_SIZE);
    raf.writeFloat(novaNota);
}


public static void mostrarEstudiant(RandomAccessFile raf, int id, String msg) throws Exception {
    System.out.println(msg + " " + id + ": " + consultarEstudiante(raf, id));
}


/**
 * Calcula la posició (offset) dins del fitxer on s'inicia el registre del videojoc amb l'ID especificat.
 *
 * @param id L'identificador del videojoc.
 * @return La posició dins del fitxer on s'inicia el registre del videojoc.
 */
private static long getSeekPosition(int id) {
    // L'operació (id - 1) serveix per obtenir un índex basat en 0.
    // (ID_SIZE + NAME_SIZE * CHAR_SIZE) calcula la mida total en bytes d'un registre de videojoc.
    // ID_SIZE representa la mida en bytes de l'ID del videojoc.
    // NAME_SIZE * CHAR_SIZE representa la mida total en bytes del nom del videojoc.
    return (id - 1) * (ID_SIZE + NAME_SIZE * CHAR_SIZE + GRADE_SIZE);
}


/**
 * Retorna una versió del nom del videojoc que sempre té una longitud fixa (NAME_SIZE).
 * Si el nom és més llarg que NAME_SIZE, es trunca. Si és més curt, s'omple amb espais en blanc.
 *
 * @param name El nom original del videojoc.
 * @return El nom amb una longitud fixa de NAME_SIZE caràcters.
 */
private static String getPaddedName(String name) {
    // Si el nom és més llarg que la mida màxima permesa (NAME_SIZE),
    // es trunca per ajustar-se a aquesta mida.
    if (name.length() > NAME_SIZE) {
        return name.substring(0, NAME_SIZE);
    }
    // Si el nom és més curt que NAME_SIZE, s'omple amb espais en blanc fins a assolir aquesta mida.
    // String.format amb "%1$-" + NAME_SIZE + "s" assegura que la cadena resultant tingui una longitud fixa.
    return String.format("%1$-" + NAME_SIZE + "s", name);
}
```