ACTIVITAT
Objectius: - Aprendre a programar objectes amb el patró ‘prototype’ en JAVA
Instruccions: - Es tracta d’un treball en grups de dos, no s’admet cap tipus de còpia. - Responeu a l’espai de cada pregunta, si ho feu amb diapositives enganxeu la diapositiva en aquest mateix espai. - Es valorarà la cura en la presentació del document i que segueixi l’estructura indicada.
Criteris d’avaluació: - Cada pregunta té el mateix pes sobre 90% - Les metodologies de treball, organització personal i participació conten un 10%
Entrega: - Aquest document amb les explicacions i captures necessàries i els arxius adjunts necessaris del codi que es demana - El nom dels arxius adjunts a entregar seràn: nomicognom-nomicognom.zip

Noms i Cognoms: Joel Berzal Álamo

Materials:

Necessiteu un entorn de desenvolupament en JAVA
Feu servir Google per buscar els tutorials que us serveixin millor
Tens més informació sobre el mètode prototype en aquest post i en aquest exemple JAVA
Creeu els arxius a la carpeta ‘src’ del projecte i executeu amb els scripts ‘./build.sh’ i ‘.\build.ps1’

Tasques:

0 - Implementa els següents objectes en JAVA, fent servir el model de clonació prototype i demostra que funciona correctament:

- Una classe Electrodomèstic amb nom, color, preu, marca, eficiència.

  Per poder crear aquesta classe, primer he hagut de crear les cinc variables públiques que s'indiquen a l'enunciat.

```java
public abstract class Electrodomestic {

    public String nom;
    public String color;
    public double preu;
    public String marca;
    public String eficiencia;

```

  A continuació, he hagut de crear dos constructors públics, dels quals un està buit i l'altre instància les cinc variables anteriors.

```java
    public Electrodomestic() {}

    public Electrodomestic(Electrodomestic target) {
        if (target != null) {
            this.nom = target.nom;
            this.color = target.color;
            this.preu = target.preu;
            this.marca = target.marca;
            this.eficiencia = target.eficiencia;
        }
    }
```

  Després, he hagut de crear un mètode públic i abstracte anomenat *clone*.

```java

    public abstract Electrodomestic clone();

```

  Finalment, he hagut de crear un booleà públic.

```java
    @Override
    public boolean equals(Object object2) {
        if (!(object2 instanceof Electrodomestic)) return false;
        Electrodomestic cast2 = (Electrodomestic) object2;
        return cast2.nom.equals(nom) && cast2.color.equals(color) && cast2.preu == preu && cast2.marca.equals(marca) && cast2.eficiencia.eq
    }
}
```

- Una classe Rentadora que és un electrodomèstic i a més té: revolucions, soroll.

  Per poder crear aquesta classe, primer he hagut de crear les dues variables públiques que s'indiquen a l'enunciat.

```java
public class Rentadora extends Electrodomestic {

    public int revolucions;
    public int soroll;

```

  A continuació, he hagut de crear dos constructors públics, dels quals un està buit i l'altre instància les dues variables anteriors.

```java
    public Rentadora() {}

    public Rentadora(Rentadora target) {
        super(target);
        if (target != null) {
            this.revolucions = target.revolucions;
            this.soroll = target.soroll;
        }
    }
```

  Després, he hagut de fer servir el mètode públic i abstracte que prèviament havia creat en la classe *Electrodomestic* per tal que retorni un nou objecte de tipus *Rentadora*.

```java

    @Override
    public Electrodomestic clone() { return new Rentadora(this); }

```

  Finalment, he hagut de crear un booleà públic.

```java
    @Override
    public boolean equals(Object object2) {
        if (!(object2 instanceof Electrodomestic) || !super.equals(object2)) return false;
        Rentadora cast2 = (Rentadora) object2;
        return cast2.revolucions == revolucions && cast2.soroll == soroll;
    }
}
```

- Una classe Nevera que és un electrodomèstic i a més té: frigories, soroll.

  Per poder crear aquesta classe, primer he hagut de crear les dues variables públiques que s'indiquen a l'enunciat.

```
public class Nevera extends Electrodomestic {

    public int frigories;
    public int soroll;

```

  A continuació, he hagut de crear dos constructors públics, dels quals un està buit i l'altre instància les dues variables anteriors.

```
    public Nevera() {}

    public Nevera(Nevera target) {
        super(target);
        if (target != null) {
            this.frigories = target.frigories;
            this.soroll = target.soroll;
        }
    }
```

  Després, he hagut de fer servir el mètode públic i abstracte que prèviament havia creat en la classe *Electrodomestic* per tal que retorni un nou objecte de tipus *Nevera*.

```

    @Override
    public Electrodomestic clone() { return new Nevera(this); }

```

  Finalment, he hagut de crear un booleà públic.

```
    @Override
    public boolean equals(Object object2) {
        if (!(object2 instanceof Electrodomestic) || !super.equals(object2)) return false;
        Nevera cast2 = (Nevera) object2;
        return cast2.frigories == frigories && cast2.soroll == soroll;
    }
}
```

- Una classe Forn que és un electrodomèstic i a més té: temperatura, autoneteja.

  Per poder crear aquesta classe, primer he hagut de crear les dues variables públiques que s'indiquen a l'enunciat.

```
public class Forn extends Electrodomestic {

    public int temperatura;
    public String autoneteja;

```

  A continuació, he hagut de crear dos constructors públics, dels quals un està buit i l'altre instància les dues variables anteriors.

```
    public Forn() {}

    public Forn(Forn target) {
        super(target);
        if (target != null) {
            this.temperatura = target.temperatura;
            this.autoneteja = target.autoneteja;
        }
    }
```

  Després, he hagut de fer servir el mètode públic i abstracte que prèviament havia creat en la classe *Electrodomestic* per tal que retorni un nou objecte de tipus *Forn*.

```

    @Override
    public Electrodomestic clone() { return new Forn(this); }

```

  Finalment, he hagut de crear un booleà públic.

```
    @Override
    public boolean equals(Object object2) {
        if (!(object2 instanceof Electrodomestic) || !super.equals(object2)) return false;
        Forn cast2 = (Forn) object2;
        return cast2.temperatura == temperatura && cast2.autoneteja.equals(autoneteja);
    }
}
```

A partir de les classes anteriors, crea una llista de instàncies per cada tipus d'electrodomèstic amb almenys 2 elements a cada llista (Rentadora, Nevera, Forn), inventa't els valors.

Project > src > Main.java

```java
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {

        List<Electrodomestic> llista = new ArrayList<>();
        List<Electrodomestic> llistaCopy = new ArrayList<>();

        Rentadora rentadora_1 = new Rentadora();
        rentadora_1.nom = "Rentadora Cecotec";
        rentadora_1.color = "Negre";
        rentadora_1.preu = 279.90;
        rentadora_1.marca = "Cecotec";
        rentadora_1.eficiencia = "D";
        rentadora_1.revolucions = 1200;
        rentadora_1.soroll = 10;
        llista.add(rentadora_1);

        Rentadora rentadora_2 = new Rentadora();
        rentadora_2.nom = "Rentadora CHiQ";
        rentadora_2.color = "Blanca";
        rentadora_2.preu = 349.00;
        rentadora_2.marca = "CHiQ";
        rentadora_2.eficiencia = "A";
        rentadora_2.revolucions = 1000;
        rentadora_2.soroll = 20;
        llista.add(rentadora_2);

        Nevera nevera_1 = new Nevera();
        nevera_1.nom = "Nevera Universalblue";
        nevera_1.color = "Blanca";
        nevera_1.preu = 547.65;
        nevera_1.marca = "UniversalBlue";
        nevera_1.eficiencia = "B";
        nevera_1.frigories = 409;
        nevera_1.soroll = 42;
        llista.add(nevera_1);

        Nevera nevera_2 = new Nevera();
        nevera_2.nom = "Nevera RT62K7005SL/ES";
        nevera_2.color = "Gris";
        nevera_2.preu = 2381.00;
        nevera_2.marca = "Samsung";
        nevera_2.eficiencia = "A";
        nevera_2.frigories = 620;
        nevera_2.soroll = 42;
        llista.add(nevera_2);

        Forn forn_1 = new Forn();
        forn_1.nom = "Forn Hisense";
        forn_1.color = "Gris";
        forn_1.preu = 269.99;
        forn_1.marca = "Hisense";
        forn_1.eficiencia = "C";
        forn_1.temperatura = 300;
        forn_1.autoneteja = "Sí";
        llista.add(forn_1);

        Forn forn_2 = new Forn();
        forn_2.nom = "Forn Bosch";
        forn_2.color = "Blau";
        forn_2.preu = 369.11;
        forn_2.marca = "Bosch";
        forn_2.eficiencia = "B";
        forn_2.temperatura = 250;
        forn_2.autoneteja = "Sí";
        llista.add(forn_2);
```

Crea després una llista on cada objecte és un clon dels objectes de la llista principal anterior.

```java

        // Clonar la llista
        for (Electrodomestic obj : llista) {
            llistaCopy.add(obj.clone());
        }

```

Fes dos bucles:

A) Compara la llista original amb ella mateixa, compara que són el mateix objecte, de la mateixa classe i tenen iguals dades.

```
75          System.out.println("Comparar la mateixa llista:\n");
76          for (int i = 0; i < llista.size(); i++) {
77              compare(i, llista.get(i), llista.get(i));
78          }
```

B) Compara la llista original amb dels clons, compara que són objectes diferents, de la mateixa classe i tenen iguals dades.

```
80          System.out.println("\n\nComparar amb la llista clonada:\n");
81          for (int i = 0; i < llista.size(); i++) {
82              compare(i, llista.get(i), llistaCopy.get(i));
83          }
84
85          System.out.println("\n\nComparar amb la llista clonada però invertida:\n");
86          for (int i = 0; i < llista.size(); i++) {
87              compare(i, llista.get(i), llistaCopy.get(llista.size() - i - 1));
88          }
89      }
```

- Les comparacions han de ser de cada element amb l'equivalent de l'altre llista (és a dir 1 a 1, no 1 a tots).