



The screenshot shows a Google Docs interface with a document titled "WS41 Quiz". The document content includes a code snippet and its output:

```
Circle: Draw() [Color : White, x : 37, y :15, radius :100  
Circle: Draw() [Color : White, x : 14, y :17, radius :100  
Circle: Draw() [Color : Blue, x : 53, y :69, radius :100  
Circle: Draw() [Color : Blue, x : 0, y :75, radius :100  
BUILD SUCCESSFUL (total time: 0 seconds)
```

41 YARANGA SUQUILLO LEONARDO JAVIER **Prototype** OK

**FAMILY: Creational design pattern family.**

**DEFINITION: It is a creational pattern that can create new objects based on an object, copying or cloning this prototype. Usually it is more used when we don't have to specify the type of the object and we need to create objects with the similar properties based on one.**

**SNIPPET:**

```
package ec.edu.espe.snippedPrototype.model;
```

```

package ec.edu.espe.snippedPrototype.model;

/**
 * @author Leonardo Yaranga, Progress Team, DCCO-ESPE
 */
public class Car implements Prototype {

    private String make;
    private String model;
    private int year;

    public Car(String make, String model, int year) {
        this.make = make;
        this.model = model;
        this.year = year;
    }

    @Override
    public Prototype clone() {
        return new Car(make, model, year);
    }

    public String getMake() {
        return make;
    }

    public String getModel() {
        return model;
    }

    public int getYear() {
        return year;
    }
}

```

```

package ec.edu.espe.snippedPrototype.model;

```

```

public static void main(String[] args) {
    Car prototype = new Car("Toyota", "Camry", 2020);
    Car copy1 = (Car) prototype.clone();
    Car copy2 = (Car) prototype.clone();
    System.out.println("copy1.getMake()"); // prints "Toyota"
    System.out.println("copy2.getYear()"); // prints 2020
}

```

EXECUTION:

```

Output - gptSnippetPrototype (run) x
>> Toyota
>> 2020
BUILD SUCCESSFUL (total time:

```

EVIDENCE:



solve a problem with prototype pattern in java



solve a problem with prototype pattern in java



Here's an example of how you can use the Prototype pattern to solve a specific problem in Java:



Suppose you want to implement a system that generates new `Car` objects with specific properties (make, model, year, etc.). You can define a `Car` class that implements the `Prototype` interface:

typescript

Copy code

```
interface Prototype {
    Prototype clone();
}

class Car implements Prototype {
    private String make;
    private String model;
    private int year;

    public Car(String make, String model, int year) {
        this.make = make;
        this.model = model;
        this.year = year;
    }

    @Override
    public Prototype clone() {
        return new Car(make, model, year);
    }

    public String getMake() {
        return make;
    }
}
```

Regenerate response

```
public String getModel() {  
    return model;  
}  
  
public int getYear() {  
    return year;  
}  
}
```

In this example, the `Car` class implements the `Prototype` interface and has a `clone` method that returns a new `Car` object with the same properties as the original object.

You can then create a `Car` object with specific properties and use the `clone` method to generate new `Car` objects with the same properties:

```
java Copy code  
  
Car prototype = new Car("Toyota", "Camry", 2020);  
Car copy1 = prototype.clone();  
Car copy2 = prototype.clone();  
  
System.out.println(copy1.getMake()); // prints "Toyota"  
System.out.println(copy2.getYear()); // prints 2020
```

This way, you can generate new `Car` objects with the same properties as the original prototype without having to write separate code for each object.

...instead of creating each object from scratch. The pattern also provides a way to abstract the creation process, so that the exact type of object being created is not known to the client code.



What is the family that prototype pattern belongs



The Prototype pattern belongs to the **Creational** pattern family.

[Regenerate response](#)



What is the family that prototype pattern belongs