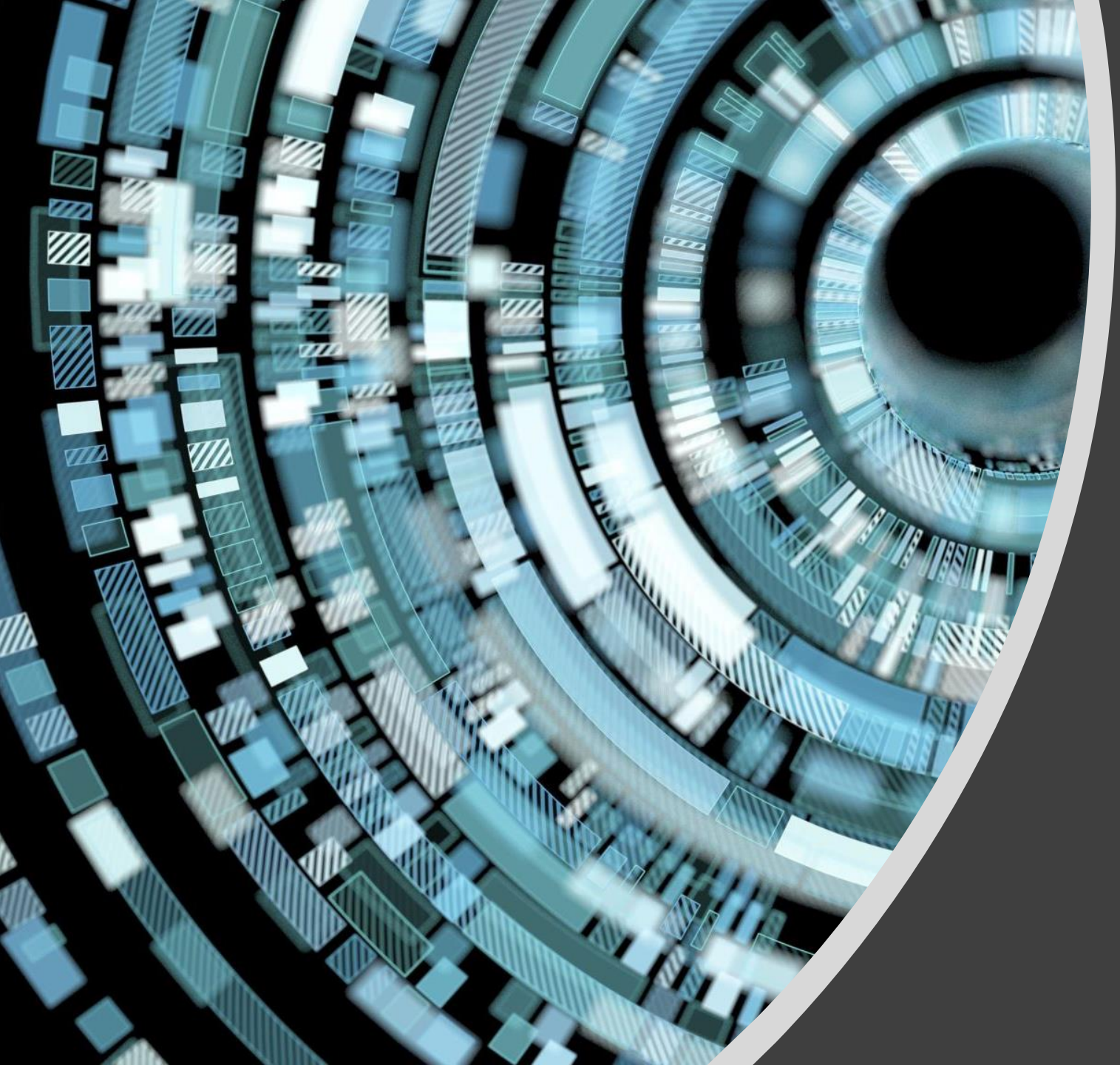
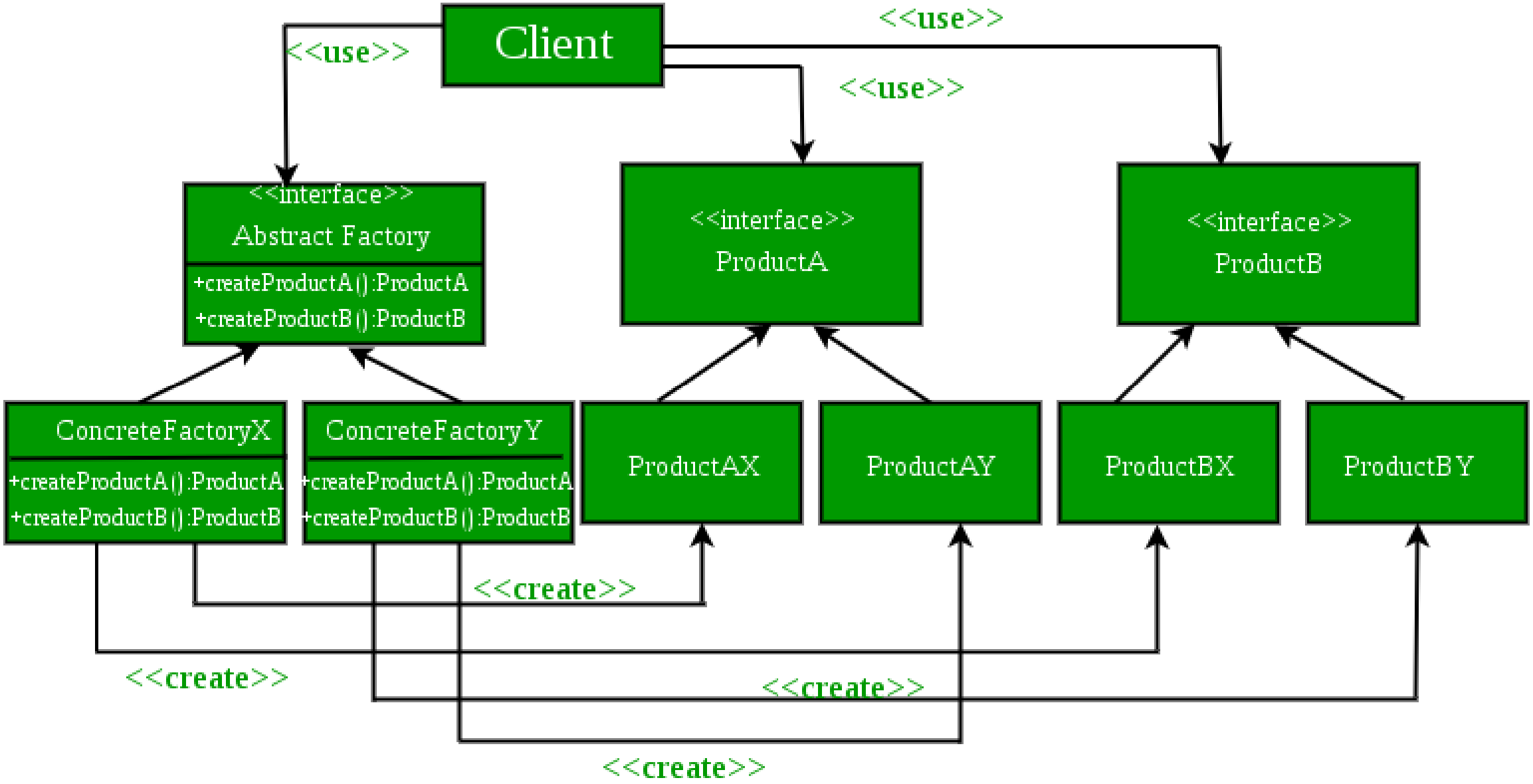


Java Abstract Factory Design Pattern

Efthimios Alepis


Client
<<use>>
<<use>>
<<use>>
<<interface>>
Abstract Factory
+createProductA():ProductA
+createProductB():ProductB
<<interface>>
ProductA
<<interface>>
ProductB
ConcreteFactoryX
+createProductA():ProductA
+createProductB():ProductB
ConcreteFactoryY
+createProductA():ProductA
+createProductB():ProductB
ProductAX
ProductAY
ProductBX
ProductBY
<<create>>
<<create>>
<<create>>
<<create>>

Remarks

- **AbstractFactory** : Declares an interface for operations that create abstract product objects.
- **ConcreteFactory** : Implements the operations declared in the AbstractFactory to create concrete product objects.
- **Product** : Defines a product object to be created by the corresponding concrete factory and implements the AbstractProduct interface.
- **Client** : Uses only interfaces declared by AbstractFactory and AbstractProduct classes.

Let’s create a full example

```java
package com.unipi.talepis;

public abstract class Phone {
    public abstract int getScreenSize();
    public abstract String getStorage();
    public abstract String getPhoneNumber();
    @Override
    public String toString(){
        return "ScreenSize= "
                +this.getScreenSize()
                +", Storage="
                +this.getStorage()
                +", PhoneNumber="
                +this.getPhoneNumber();
    }
}
```

```java
package com.unipi.talepis;

public class FeaturePhone extends Phone{
    private final int screenSize;
    private final String storage;
    private final String phoneNumber;

    public FeaturePhone(int screenSize, String storage, String phoneNumber) {
        this.screenSize = screenSize;
        this.storage = storage;
        this.phoneNumber = phoneNumber;
    }

    @Override
    public int getScreenSize() { return screenSize; }

    @Override
    public String getStorage() { return storage; }

    @Override
    public String getPhoneNumber() { return phoneNumber; }
}
```

```java
package com.unipi.talepis;

public class SmartPhone extends Phone{
    private final int screenSize;
    private final String storage;
    private final String phoneNumber;

    public SmartPhone(int screenSize, String storage, String phoneNumber) {
        this.screenSize = screenSize;
        this.storage = storage;
        this.phoneNumber = phoneNumber;
    }

    @Override
    public int getScreenSize() { return screenSize; }

    @Override
    public String getStorage() { return storage; }

    @Override
    public String getPhoneNumber() { return phoneNumber; }
}
```

```java
package com.unipi.talepis;

public interface PhoneAbstractFactory {
    Phone createPhone();
}
```

```java
package com.unipi.talepis;

public class FeaturePhoneFactory implements PhoneAbstractFactory{
    private final int screenSize;
    private final String storage;
    private final String phoneNumber;

    public FeaturePhoneFactory(int screenSize, String storage, String phoneNumber) {
        this.screenSize = screenSize;
        this.storage = storage;
        this.phoneNumber = phoneNumber;
    }

    @Override
    public Phone createPhone() { return new FeaturePhone(screenSize,storage,phoneNumber); }
}
```

```java
package com.unipi.talepis;

public class SmartPhoneFactory implements PhoneAbstractFactory{
    private final int screenSize;
    private final String storage;
    private final String phoneNumber;

    public SmartPhoneFactory(int screenSize, String storage, String phoneNumber) {
        this.screenSize = screenSize;
        this.storage = storage;
        this.phoneNumber = phoneNumber;
    }

    @Override
    public Phone createPhone() {
        return new SmartPhone(screenSize,storage,phoneNumber);
    }
}
```

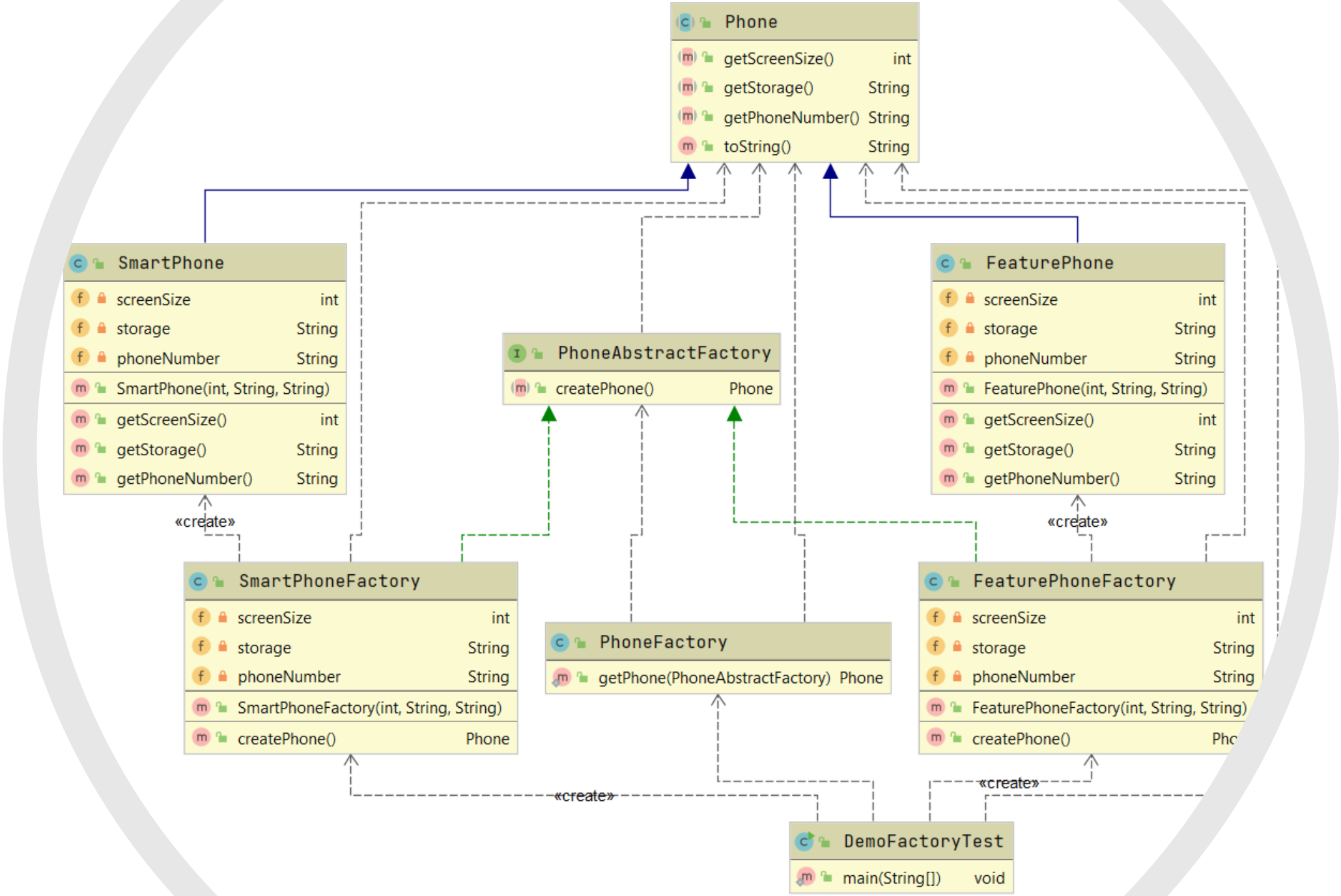
```java
package com.unipi.talepis;

public class PhoneFactory {
    public static Phone getPhone(PhoneAbstractFactory factory){
        return factory.createPhone();
    }
}
```

```java
package com.unipi.talepis;

public class DemoFactoryTest {

    public static void main(String[] args) {
        Phone fPhone1 = PhoneFactory.getPhone(new FeaturePhoneFactory( screenSize: 2, storage: "25", phoneNumber: "+306543"));
        Phone smPhone1 = PhoneFactory.getPhone(new SmartPhoneFactory( screenSize: 10, storage: "32", phoneNumber: "+3076543"));
        System.out.println("Feature Phone1:"+fPhone1.toString());
        System.out.println("SmartPhone1:"+smPhone1.toString());
    }
}
```


Phone
getScreenSize() int
getStorage() String
getPhoneNumber() String
toString() String
SmartPhone
screenSize int
storage String
phoneNumber String
SmartPhone(int, String, String)
getScreenSize() int
getStorage() String
getPhoneNumber() String
PhoneAbstractFactory
createPhone() Phone
FeaturePhone
screenSize int
storage String
phoneNumber String
FeaturePhone(int, String, String)
getScreenSize() int
getStorage() String
getPhoneNumber() String
«create»
«create»
SmartPhoneFactory
screenSize int
storage String
phoneNumber String
SmartPhoneFactory(int, String, String)
createPhone() Phone
PhoneFactory
getPhone(PhoneAbstractFactory) Phone
FeaturePhoneFactory
screenSize int
storage String
phoneNumber String
FeaturePhoneFactory(int, String, String)
createPhone() Pho
«create»
«create»
DemoFactoryTest
main(String[]) void