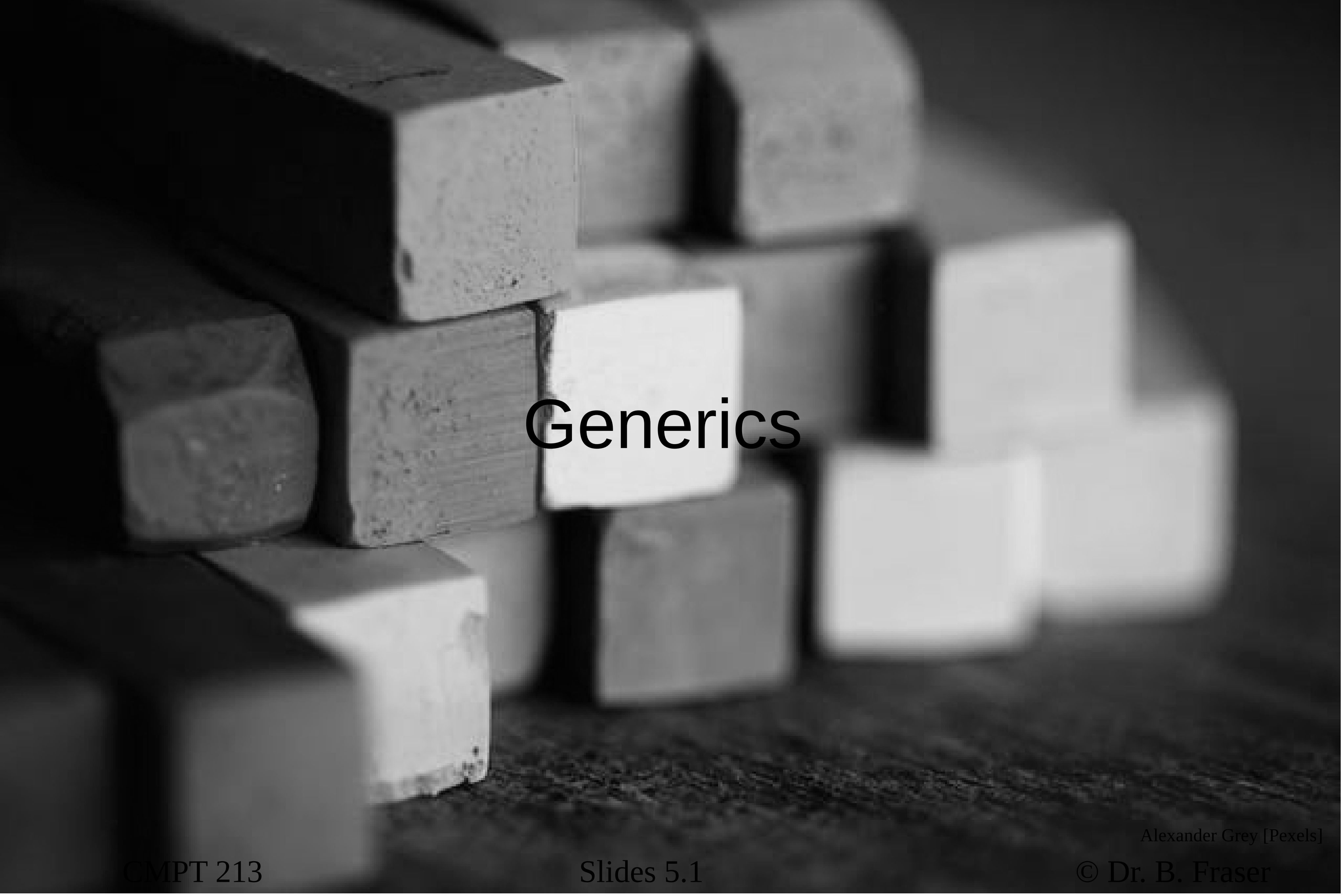

Generics

Alexander Grey [Pexels]

Generics

- Generic Type Examples

  ArrayList<Car>

  ArrayList<Fruit>
- ..
  - Generics give Java code

    ..
  - Code is written once, but handles different types. Selection is done at compile-time.
- It’s different than Runtime Polymorphism
  - ..   gives runtime polymorphism
  - Code is written once, but handles different types. Selection is done at run-time.

Generics and Different Types

- Generics handle any object type
  - Code written with a generic can handle **any** type of object, not just ones related via inheritance.
  - The same ArrayList code can make:
    - an ArrayList of Cars, or
    - an ArrayList of Fruit,
    - ...
- Once created, an object of type ArrayList<Car> cannot handle Fruit:
  - An ArrayList<Car> object..

```
ArrayList<Car> myCars = new ArrayList<>();
Car firstCar = myCars.get(0);
```

Generic Method

- Generic Method
  - A method which has a..
  - It can use this type parameter as a regular type
- Can call a generic method with any type of object
  - Compiler ensures that it preserves the type

T is the
type parameter

```
    public static <T> List<T> makeIntoList(T obj1, T obj2)  {
        List<T> myList = new ArrayList<>();
        myList.add(obj1);
        myList.add(obj2);
        return myList;
    }
```

Generic Method Example

```java
public class GenericMethod {

    public static <T> List<T> makeIntoList(T obj1, T obj2){
        List<T> myList = new ArrayList<>();
        myList.add(obj1);
        myList.add(obj2);
        return myList;
    }

    public static void main(String[] args) {
        // Call makeIntoList() on Strings
        List<String> myStrings = makeIntoList("Hello", "World");

        // Call makeIntoList() on Cars
        Car car1 = new Car("Forester", 2050);
        Car car2 = new Car("Model T", 1920);
        List<Car> myCars = makeIntoList(car1, car2);
    }
}
```

Generic Class

- Generic Classes
  have a type parameter for the whole class

```
public class ShippingCrate<T> {
    private T item;

    public ShippingCrate(T item) {
        this.item = item;
    }

    public T getItem() {
        return item;
    }

    public void printLabel() {
        System.out.println("One shipping crate containing: ");
        System.out.println("     " + item.toString());
    }
}
```

Generic Interfaces

- Generic Interfaces
  - Like a class, has a type parameter for the whole interface.
  - Very useful to make flexible code

- Can use
  ..
  for client code to provide an implementation which fills in a part of our algorithm.
- Our object is then typed to the type the client needs.

```
// Create an object that, given an item,
// provides the description you want.
public interface Describer<T> {
    String getDescription(T item);
}
```

Summary

- Generic
  - Provides compile-time polymorghism
- Inheritance
  - Provides run-time polymorphism
- Generic methods
  Written once, work on any (specific) type of object
- Generic class
  Handle any (specific) type of object
- Generic interface
  Provides flexible ability to the strategy pattern