159 XTENDED

A WEEKLY REVIEW

GENERICS

Pull up **Socrative.com**

and join room **XTEND159**

THINKING ABOUT ARRAYS...

- When creating an array, we must specify it's **type**.
- The type refers to what kind of thing we want to store in the array.
- We can declare an array of any primitive types or reference types we want.
- All arrays are declared the same way.
- **We don't need different types of arrays for different types of data.**

```
int[] numbers = new int[10];
float[] totals = new float[99];
String[] names = new String[4];
Square[] windows = new Square[7];
```

DATA TYPES IN CLASSES

- Here, we have a basic class called CarGarage.
- It's only purpose is to store two Car objects.
- It can ONLY store car objects.
- When we want to store something else…

```
public class CarGarage
{
    public Car carOne;
    public Car carTwo;

    public CarGarage(Car carOne, Car carTwo)
    {
        this.carOne = carOne;
        this.carTwo = carTwo;
    }
}
```

DATA TYPES IN CLASSES

- Here, we have a basic class called CarGarage.
- It's only purpose is to store two Car objects.
- It can ONLY store car objects.
- When we want to store something else…
- We have to create another entire class.
- Wouldn't it be nice if Garage type worked like Array types?

```
public class CarGarage
{
    public Car carOne;
    public Car carTwo;

    public CarGarage(Car carOne, Car carTwo)
    {
        this.carOne = carOne;
        this.carTwo = carTwo;
    }
}

public class BoatGarage
{
    public Boat boatOne;
    public Boat boatTwo;

    public BoatGarage(Boat boatOne,
      Boat boatTwo)
    {
        this.boatOne = boatOne;
        this.boatTwo = boatTwo;
    }
}
```

GENERIC TYPES

- **We can add a new parameter for data types.**
- We add two triangle brackets on the end of the class name with a new variable inside of it.
- We can now use this variable to **make our data types generic**.

```
public class GenericGarage<E>
{
    public Car carOne;
    public Car carTwo;

    public GenericGarage(Car carOne, Car carTwo)
    {
        this.carOne = carOne;
        this.carTwo = carTwo;
    }
}
```

GENERIC TYPES

- Inside of our class, we use the new variable in place of a data type.
- Type E is our new generic type.

```
public class GenericGarage<E>
{
    public E thingOne;
    public E thingTwo;

    public GenericGarage(E thingOne, E thingTwo)
    {
        this.thingOne = thingOne;
        this.thingTwo = thingTwo;
    }
}
```

GENERIC TYPES

- Inside of our class, we use the new variable in place of a data type.
- Type E is our new generic type.
- Now, when someone creates a GenericGarage, they specify a data type in triangle brackets.
- Every variable E in the class is changed to this specified type.

```
GenericGarage<Boat> = new GenericGarage<Boat>( … );
```

```
public class GenericGarage<E>
{
    public E thingOne;
    public E thingTwo;

    public GenericGarage(E thingOne, E thingTwo)
    {
        this.thingOne = thingOne;
        this.thingTwo = thingTwo;
    }
}
```

```
public class GenericGarage<Boat>
{
    public Boat thingOne;
    public Boat thingTwo;

    public GenericGarage(Boat thingOne, Boat thingTwo)
    {
        this.thingOne = thingOne;
        this.thingTwo = thingTwo;
    }
}
```

ARRAYLISTS

- You have already been using generic types with ArrayLists.
- When declaring an ArrayList, you always specify a type in triangle brackets.
- This is because the ArrayList class is **generically typed**.
- **Using generics makes your classes more versatile and reusable.**

```
ArrayList<int> numbers =
  new ArrayList<int>;

ArrayList<String> names =
  new ArrayList<String>;
```