Java - <Generics>

Dr. Susan McKeever

Using generics in Java

- Generics were introduced in Java 5 (relatively new)
- ***Java Generics is a language feature of Java that allows for the definition and use of generic types and methods.*** ***Eh?***
- Purpose is best explained through examples
- Note syntax: < > used to declare a generic type.
- e.g. <e> or <T> are often used

e stands for element, T for type..

Case 1: Overloading method due to different types to be supported

- printing different types of arrays..
- Multiple overloaded methods versus one "generic" method – see printArray() example
- Use a **generic method**

Case 2 – A class that needs generics

- Class has an attribute.. and it's easy to cause a run time error - see example.
- Solution.. just change the attribute to Integer?.. but then Space can only store on type of thing..
- Better solution – use generic type (class)
  – Same error is detected at compile time
  – much safer

Note syntax for implementing..

```
SaferSpace<Integer> mySpace = new SaferSpace<Integer>();
mySpace.add(60);
//mySpace.add("desk");

Integer i = mySpace.get();
System.out.println(i);
```

```
public class SaferSpace<T>
{
	private T furnitureID;

	public void add(T t)
	{
		this.furnitureID = t;
	}
```

And Note....

- You can use any name you want for the generic parameter - as long as < > used

```
public class SaferSpace<FurnitureItem>
{
	private FurnitureItem furnitureID;

	public void add(FurnitureItem item)
	{
		this.furnitureID = item;
	}
```

(but T or e is most often used)

Generic class

- A generic class is defined with the following format:

```
class name<T1, T2, ..., Tn> {
 /* ... */ }
```

Bounded types

- "Bounded" here means having restrictions.
  - = using generic types that have some sort of limit on their allowable types

- E.g. allow a generic type, as along as it is a subclass of class Shape (or implements interface Shape!)

```
public class SaferSpace<T extends Shape>
{
	private T furnitureID;

	public void add(T t)
	{
		this.furnitureID = t;
	}
```

Why use Generics?

Reasons to use Generics -

- Stronger type checking at compile time.
- Elimination of explicit cast.
- Enabling better code reusability such as implementation of generic algorithms

We see them all over the API – particularly in Collections

Some other parts (e.g. wildcards ?) not covered -