Object Oriented Programming		Discover Your Furthers
	Generics	
	Lecture – 7	
		SLIIT - Faculty of Computing

ООР	Contents	SLIIT Discover Your Future
	Wrapper classes	
	• Introduction to Generics	
	Generic classes - Declaration & Instantiation	
	• Generic Methods – Implementation & Invocat	ion
	Bounded Type Parameters	
	• Wildcards	
	Limitations in Generics	
	SLIIT - Fac	ulty of Computing

ООР				S. Discover	Your Future
	Wra	appe	r cla	ass / Covering class	
				n provides the mechanism to convert to object and object into primitive type.	_
		ach prim lang pac		tatype, there exists a covering class in the	
		byte	\rightarrow	Byte	
		short	\rightarrow	Short	
		int	\rightarrow	Integer	
		long	\rightarrow	Long	
		float	\rightarrow	Float	
		double	\rightarrow	Double	
		char	\rightarrow	Character	
		Boolear	1→	Boolean	3
				SLIIT - Faculty of Co	mputing

Wrapper class cont. • The automatic conversion of primitive type into object is known as autoboxing and vice-versa unboxing. • Since J2SE 5.0, autoboxing and unboxing feature converts primitive into object and object into primitive automatically. | public class Test { | public class Test { | public static void main(String args[]) { | int a = 50; | int a = 50; | Integer | = a; //auto boxing | system.out.println(a); | system.out.println(i); | system.out.println(j); | autoboxing.java | | SLIIT - Faculty of Computing

Concept of "generics" was introduced in JDK 5 to deal with type-safe objects. Introduction of generics has changed Java in 2 ways: Added new syntax to Java language Caused changes to many of the classes & methods in the core API With the use of generics, it is possible to create classes, interfaces & methods which works in type-safe manner.

SLIIT

Generics in Java cont.

- "Generics" means "parameterized types".
- Many algorithms are same regardless of its data type that is applied. With generics, you can define the algorithm once independently of any specific type of data. Later, you can use the algorithm to a wide variety of data types without any additional effort.

SLIIT - Faculty of Computin

SLIIT

Advantages of Generics

- Type-safety: We can hold only a single type of objects in generics. It doesn't allow to store other types of objects.
- Type casting is not required: There is no need to typecast the object. All type conversions are implicit.
- 3. Compile-Time Checking: It is checked at compile time so problem will not occur at runtime. The good programming strategy says it is far better to handle the problem at compile time than at runtime.

SLIIT - Faculty of Computing

SLIIT

Implementation

• When you are in need to store a list of values, you shall use an array.

e.g.: if you are storing marks of 5 students then you may create an array as shown in the code below $\,$

1	
2	<pre>public class GenExample {</pre>
30	<pre>public static void main(String args[]){</pre>
4	<pre>int marks[] = new int[5];</pre>
5	}
6	}

SLIIT - Faculty of Computi

Implementation cont.

Problem:

- Size of the array is fixed. If there is a change in the size of the array (in the number of elements that are stored), you have to modify the code manually.
- You may not be able to expand or shrink the array automatically as & when the element is being added.

Solution:

• Use Collection Interface

SLIIT - Faculty of Computin

GenericDemo2.java

SLIIT - Faculty of Computing

SLIIT - Faculty of Comp

SLIIT

Implementation cont.

Shown below is an example of how Collection is used to store elements. Note that the list grows each & every time add() is called.

| 10 import java.util.ArrayList; | 2 import java.util.ArrayList; | 2 import java.util.Collection; | 3 | 4 | public class GenExample { | 50 | public static void main(String args[]) { | 6 | Collection value = new ArrayList(); | 7 | value.add(10); | 7 | value.add(20); | 9 | value.add(30); | 10 | value.add(40); | 11 | value.add(50); | 12 | value.add(50); | 13 | value.add(50); | 14 | value.add(50); | 15 | value.add(50); | 16 | value.add(50); | 17 | value.add(50); | 18 | value.add(50); | 19 | value.add(50); | 10 | value.add(50);

Implementation cont. Collection value = new ArrayList(); This statement allows you to create a list of elements (list of marks). Note that Collection is an interface and cannot be instantiated directly. ArrayList is a class which implements the Interface List which extends the Interface Collection.

1

Implementation cont. Collection value = new ArrayList(); Problem: This list may contain any object as we have not specified the data type of the element to be added. Collection value = new ArrayList(); value.add("SLIIT"); //String object value.add(145); //integer value.add(23.4578); //double value.add(1.54f); //float value.add('y'); //char

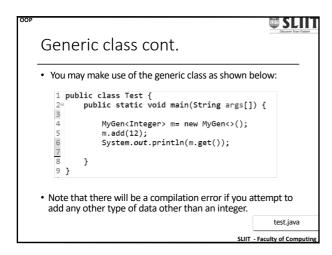
Implementation cont. Collection value = new ArrayList(); Problem: What if the requirement is to store only integers? Solution: Make use of Generics! Collection <Integer> value = new ArrayList<>(); *Make use of Wrappers as primitive types do not support Generics.

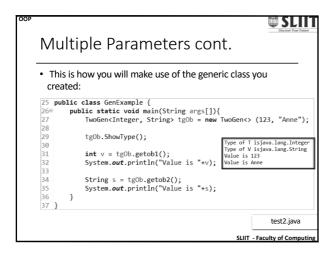
o 7 value.add("SLIIT"); //String object 8 value.add(15); //integer 9 value.add(23.4578); //double	,One cer	ion <integer> value = new ArrayList<>(</integer>
<pre>2 import java.util.Collection; 3 4 public class GenExample { 5 public static void main(String args[]){ 6 Collection<integer> value = new ArrayList<>(); 7 value.add("SLIIT"); //String object 8 value.add(145); //integer 9 value.add(23.4578); //double</integer></pre>		
<pre>value.add(1.54f); //float value.add('y'); //char</pre>	2 3 4 5 6 7 8	<pre>public class GenExample { public static void main(String args[]){ Collection</pre> // Collection // value.add("SLIIT"); //String object // value.add(145); //integer // value.add(23.4578); //double

OOP		SLIIT Discover Your Future
	Question:	
	Write a program to store a list of names, retrievames & display on the screen.	ve the
	Hint: Use an ArrayList class	
	SUIT - Fac	16 ulty of Computing
	JEHT TOU	, c. companing

ООР		SLIIT Discover Your Future
	Type Parameters	
	The type parameter naming conventions are as follows:	;
	T - Type E - Element	
	K - Key	
	N - Number	
	V - Value	
		17
1	SLIIT - Fac	ulty of Computing

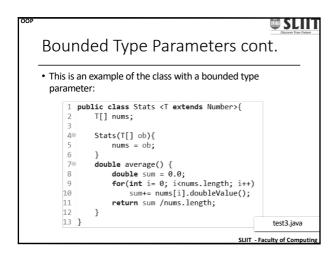
ООР		SLIIT Discover Your Future
	Generic class	
	 A class that can refer to any type is known as gene Type T indicates that it can refer to any type (Integrouble, Employee etc.) 	
	<pre>public class MyGen<t> { T obj; void add(T val) { this.obj = val; }</t></pre>	
	7 } 8 9 T get() { 10 return obj; 11 } 12 }	test.java
	SLIIT	Faculty of Computing





Multiple Parameters cont. • Although the two type argument differ in the example, it is possible for both types to be same. e.g.: TwoGen<String, String> tg0b = new TwoGen<> ("BM-1", "Anne"); tg0b.ShowType(); Etring v = tg0b.getob1(); System.out.println("Value is "+v); Walue is BM-1 Value is Anne String s = tg0b.getob2(); System.out.println("Value is "+s); • Also note, if both type arguments were always the same, then two type parameters would be unnecessary. 23 SUIT. Faculty of Computing.

OOP		SLII T Discover Your Future
	Bounded Type Parameters	
	There may be times when you will want to <i>restrict</i> the k of data types that are allowed to be passed to a type parameter.	ind
	e.g.: A method that operates on numbers might only want accept instances of Number or its subclasses.	to
	 To declare a bounded type parameter, list the type parameter's name, followed by the extends keyword, followed by its upper bound in the following format. 	
	<t extends="" superclass=""></t>	
	e.g.: <t extends="" number=""></t>	
	Note: T, can only be replaced by the superclass or its subclasses.	
		24
	SLIIT - Faculty	of Computing



OOP	SL Dissays biar	Future
	Bounded Type Parameters cont.	
	Note that the following code segment will give compilation error as String is not a subclass of Number	
	String snum[] = {"1", "2", "3", "4", "5"}; Stats <u>String</u> ob3 = new Stats<>(<u>snums</u>);	
	27	
	SLIIT - Faculty of Com	uting

Limitations in Generics

- Type parameters cannot be instantiated It is not possible to create instances of a type parameter
- Restriction on static members No static member can use a type parameter declared by the enclosing class. Note that you can declare static generic methods.
- Generic array restriction -
 - Cannot instantiate an array whose element type is a type parameter.

 • Cannot create an array of type specific references
- Generic exception restriction A generic class cannot extend
 Throwable. This means we cannot create generic exception classes.

SLIIT