


# Generics



Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

---

---

---

---

---

---

---

---

## Generic Types in Java

```
public class Pair< T >
```

— type variable (type parameter)

— formal type parameter list

— generic (parameterized) type

type declaration

```
Pair< Integer > intPair;
```

— actual type to map to T

— actual type parameter list

type instantiation

Format for a generic (parameterized) type and instantiation of a generic type

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

---

---

---

---

---

---

---

---

## Using the Pair<T> class

```

/**
 * Illustrate the use of a generic type.
 */
public class GenericPair {
    public static void main(String[] args) {
        Pair<String> stringPair = new Pair<String>("string", "Pair");
        Pair<Integer> intPair = new Pair<Integer>(1, 2);

        System.out.println("intPair is: " + intPair);
        System.out.println("stringPair is: " + stringPair);

        intPair.swapElements();
        System.out.println("After swapping elements, intPair is " + intPair);
        intPair.setFirstElement(new Integer(-1));
        stringPair.setSecondElement("Generic types are useful");

        System.out.println("The pairs after some resetting:");
        System.out.println("intPair is: " + intPair);
        System.out.println("stringPair is: " + stringPair);
        Integer intElement = intPair.getFirstElement();
        String stringElement = stringPair.getFirstElement();

        System.out.println("intElement is " + intElement +
            " and stringElement is " + stringElement);
    }
}

```

Specifying the type parameter

stringPair can store a pair of String objects

intPair can store a pair of Integer objects

Program Output:

```

intPair is: < 1, 2 >
stringPair is: < string, Pair >
After swapping elements, intPair is < 2, 1 >
The pairs after some resetting:
intPair is: < -1, 1 >
stringPair is: < string, Generic types are useful
intElement is -1 and stringElement is string

```

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

---

---

---

---

---

---

---

---



### Generic & Erasure: The Client Class

```

/**
 * Illustrate use of a generic type. The client class after erasure.
 */
public class GenericEr {
    public static void main ( String args[] ) {
        Pair stringPair = new Pair( "string", "Pair" );
        Pair intPair = new Pair( 1, 2 );

        System.out.println( "intPair is: " + intPair );
        System.out.println( "stringPair is: " + stringPair );

        intPair.swapElements();
        System.out.println( "\nAfter swapping elements, intPair is "
            + intPair );
        intPair.setFirstElement( new Integer( -1 ) );
        stringPair.setSecondElement( "Generic types are useful" );

        System.out.println( "\nThe pairs after some resetting: " );
        System.out.println( "intPair is: " + intPair );
        System.out.println( "\tstringPair is: " + stringPair );
        Integer intElement1 = (Integer)intPair.getFirstElement();
        String stringElement1 = (String)stringPair.getFirstElement();

        System.out.println( "\nintElement1 is " + intElement1 +
            " and stringElement1 is " + stringElement1 );
    }
}

```

All occurrences of the formal type are removed

Remember, the *compiler* takes care of doing erasure

Type casts are need to convert the *Object* references returned by the get methods to their respective types; the *compiler* does this for you!

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

---

---

---

---

---

---

---

---

### Enhanced Looping

```

List<Integer> list = new
    LinkedList<Integer>();
for (Integer i : list) { ... }

```

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

---

---

---

---

---

---

---

---

### Autoboxing/Auto-Unboxing

```

ArrayList<Integer> list = new ArrayList<Integer>();
list.add(0, 42);
int total = list.get(0);

```

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

---

---

---

---

---

---

---

---

## Enumerated Types

```
public class Card {
    public enum Rank { DEUCE, THREE, FOUR, FIVE, SIX,
                     SEVEN, EIGHT, NINE, TEN, JACK, QUEEN, KING, ACE }
    public enum Suit { CLUBS, DIAMONDS, HEARTS, SPADES }

    private final Rank rank;
    private final Suit suit;
    private Card(Rank rank, Suit suit) {
        this.rank = rank;
        this.suit = suit;
    }

    public Rank rank() { return rank; }
    public Suit suit() { return suit; }
    public String toString() { return rank + " of " + suit; }

    private static final List<Card> protoDeck = new ArrayList<Card>();

    // Initialize prototype deck
    static {
        for (Suit suit : Suit.values())
            for (Rank rank : Rank.values())
                protoDeck.add(new Card(rank, suit));
    }

    public static ArrayList<Card> newDeck() {
        return new ArrayList<Card>(protoDeck);
    }
}
```

Copyright © 2004, Addison Wesley Longman, Inc. All rights reserved. Return copy of prototype deck

---

---

---

---

---

---

---

---