**2.** Will the following method compile? If not, why?

public static void print(List list)

{

for (Number n : list)

System.out.print(n + " ");

System.out.println();

}

**Answer:**

Yes, this method is upperbound wildcard.

**3.** If the compiler erases all type parameters at compile time, why should you use generics?

**Answer:**

Generics promote code reusability

Individual typecasting is not required

Generics support programming types as parameters.

**4.** What is the following class converted to after type erasure?

public class Pair <K, V> {

public Pair(K key, V value) {

this.key = key;

this.value = value;

}

public K getKey() { return key; }

public V getValue() { return value; }

public void setKey(K key) { this.key = key; }

public void setValue(V value) { this.value = value; }

private K key;

private V value;

}

**Answer:**

public class Pair {

public Pair(Object key, Object value) {

this.key = key;

this.value = value;

}

public Object getKey() { return key; }

public Object getValue() { return value; }

public void setKey(Object key) { this.key = key; }

public void setValue(Object value) { this.value = value; }

private Object key;

private Object value;

**}**

**5.** Look at the following code snippet and select the correct option:

class Test extends Exception { }

class Main {

public static void main(String args[]) {

try {

throw new Test();

} catch(Test t) {

System.out.println("Got the Test Exception");

}

finally {

System.out.println("Inside finally block ");

}

}

}

a) Got the Test Exception

Inside finally block

b) Got the Test Exception

c) Inside finally block

d) Compiler Error

**Answer:**

a) Got the Test Exception

Inside finally block

**6.** What will be the output of the following code:

class Test {

public static void main(String[] args) {

try {

int a[]= {1, 2, 3, 4};

for (int i = 1; i <= 4; i++) {

System.out.println ("a[" + i + "]=" + a[i] + "\n");

}

} catch (Exception e) {

System.out.println ("error = " + e);

}

catch (ArrayIndexOutOfBoundsException e) {

System.out.println ("ArrayIndexOutOfBoundsException");

}

}

}

a) Compiler error

b) Run time error

c) ArrayIndexOutOfBoundsException

d) Error Code is printed

e) Array is printed

**Answer:**

a) Compiler error