/\*

1. What are the advantages of using generic classes?

**Answer:** Generics increase the reusability of the code, making it so that you don’t need to write code to handle different data types. Generics are also type-safe, and have a performance advantage because it removes the possibilities of boxing and unboxing.

2. Create a generic class named Triple that has 3 type

parameters and a variable for each type parameter.

Create a constructor that sets those variables.

Create get and set methods for each variable.

**Answer:**

public class Triple{  
public class Prize{  
 private String Prize;  
  
public String getName() {  
 return Prize;  
 }  
public void setName(String newName) {  
 this.Prize = newName;  
 }   
}  
 public void main(String[] args){  
 Prize box1 = new Prize();  
 Prize box2 = new Prize();  
 Prize box3 = new Prize();  
   
 box1.setName("Money");  
 box2.setName("Gun");  
 box3.setName("Flowers");  
   
 System.out.println(box1.getName());  
 System.out.println(box2.getName());  
 System.out.println(box3.getName());  
}  
}

3 (Distinct elements in ArrayList) Write the following

method that returns a new ArrayList. The new list contains

the nonduplicate elements from the original list.

public static ArrayList removeDuplicates(ArrayList list)

**Answer:**

public class Question\_3 {

public static void main(String[] args) {

ArrayList<String> strings = new ArrayList<>();

for (int i = 0; i < 10; i++) {

strings.add("Hello");

}

strings = removeDuplicates(strings);

for (String s : strings) {

System.out.println(s);

}

}

public static <E> ArrayList<E> removeDuplicates(ArrayList<E> list) {

ArrayList<E> newList = new ArrayList<>(list.size());

for (E aList : list) {

if (!newList.contains(aList)) {

newList.add(aList);

}

}

return newList;

}

}

4 (Generic linear search) Implement the following generic

method for linear search:

public static > int linearSearch(E[] list, E key)

**Answer:**

public class Exercise\_04 {

public static void main(String[] args) {

Integer[] list = new Integer[20];

for (int i = 0; i < list.length; i++) {

list[i] = i;

}

System.out.println(linearSearch(list, 1));

System.out.println(linearSearch(list, 10));

System.out.println(linearSearch(list, 30));

}

public static <E extends Comparable<E>> int linearSearch(E[] list, E value) {

for (int i = 0; i < list.length; i++) {

if (list[i].compareTo(value) == 0) {

return i;

}

}

return -1;

}

}

5. Fix all errors:

public Pair

{

A first;

B second;

public Pair<A, B>(A a, B b)

{

a = first;

b = second;

}

public void set(a, b)

{

first = a;

second = b;

}

public A getFirst()

{

return first;

}

public B getSecond()

{

return second;

}

}

**Corrected Version:**

public class Pair<A, B>{   
   
 private A first;  
 private B second;  
   
 public Pair(A first, B second) {  
 this.first = first;  
 this.second = second;  
 }  
 public void set(A first){  
 this.first = first;  
 }  
 public void setSecond(B second) {  
 this.second = second;  
 }  
 public A getFirst()  
 {  
 return first;  
 }  
 public B getSecond()  
 {  
 return second;  
 }  
}

6.

a. Why should a class such as Rectangle not be generic?

It should not be generic, as size, field and object data stored in the generic class may be lost. Generics are compile-time only operation, and you cannot declare static fields of an operation.

b. Consider the following code:

class Rectangle<A, B>

{

A length;

B width;

public Rectangle(A l, B w)

{

length = l;

width = w;

}

public double area()

{

return length \* width;

}

}

Are you allowed to do the following without errors:

Rectangle<Integer, Integer> r = new Rectangle<Integer, Integer>(5, 7);

System.out.println(r.area());

\*/

public class HW2\_Generics{

public static void main(String[] args){

}

}

Answer: No, there are errors when running the code.