# Day 19

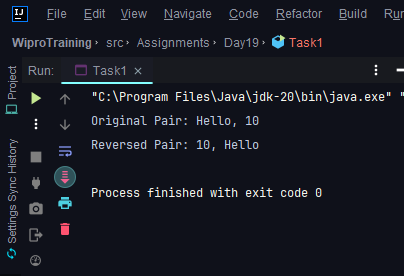
**Task 1: Generics and Type Safety**

Create a generic Pair class that holds two objects of different types, and write a method to return a reversed version of the pair.

**Program:**

package Assignments.Day19;  
  
public class Task1<S, I> {  
 private final S first;  
 private final I second;  
  
 public Task1(S first, I second){  
 this.first = first;  
 this.second = second;  
 }  
  
 public S getFirst() {  
 return first;  
 }  
  
 public I getSecond() {  
 return second;  
 }  
 private Task1<I,S> reverseOrder(){  
 return new Task1<>(second,first);  
 }  
  
 public static void main(String[] args) {  
 Task1<String, Integer> originalPair = new Task1<>("Hello",10);  
 Task1<Integer, String> reversedPair = originalPair.reverseOrder();  
   
 System.out.println("Original Pair: " + originalPair.getFirst() + ", " + originalPair.getSecond());  
 System.out.println("Reversed Pair: " + reversedPair.getFirst() + ", " + reversedPair.getSecond());  
 }  
}

**Output:**



**Task 2: Generic Classes and Methods**

Implement a generic method that swaps the positions of two elements in an array, regardless of their type, and demonstrate its usage with different object types.

**Program:**

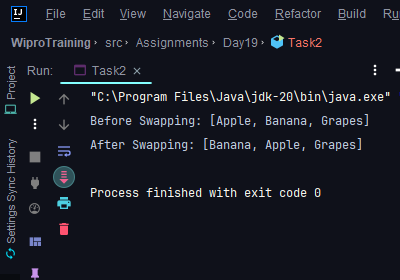
package Assignments.Day19;  
  
import java.util.Arrays;  
  
  
public class Task2 {

public static <Temp> void swap(Temp[] arr, int i, int j) {  
 Temp temp = arr[i];  
 arr[i] = arr[j];  
 arr[j] = temp;  
 }  
  
 public static void main(String[] args) {

String [] arr = {"Apple", "Banana","Grapes"};  
 System.out.println("Array:"+Arrays.toString(arr));  
 swap(arr, 0, 1);  
 System.out.println("After Swapping:"+Arrays.toString(arr));

}  
}

**Output:**



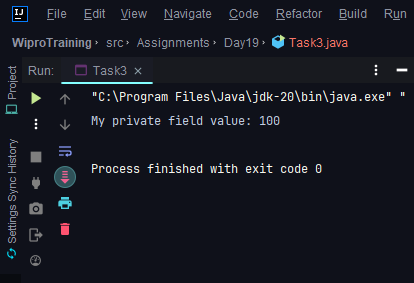
**Task 3: Reflection API**

Use reflection to inspect a class's methods, fields, and constructors, and modify the access level of a private field, setting its value during runtime.

**Program:**

package Assignments.Day19;  
import java.lang.reflect.Field;  
  
class MyClass {  
 private int myPrivateField = 42;  
  
 public void printValue() {  
 System.out.println("My private field value: " + myPrivateField);  
 }  
}  
  
public class Task3 {  
 public static void main(String[] args) throws NoSuchFieldException, IllegalAccessException {  
 MyClass obj = new MyClass();  
  
  
 Class<?> clazz = obj.getClass();  
 Field privateField = clazz.getDeclaredField("myPrivateField");  
  
 privateField.setAccessible(true);  
  
 // Modify the field value  
 privateField.setInt(obj, 100);  
  
 // Verify the updated value  
 obj.printValue();  
 }  
}

**Output:**



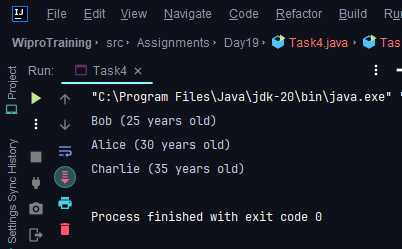
**Task 4: Lambda Expressions**

Implement a Comparator for a Person class using a lambda expression, and sort a list of Person objects by their age.

**Pogram:**

package Assignments.Day19;  
  
import java.util.ArrayList;  
import java.util.Comparator;  
import java.util.List;  
  
class Person {  
 private final String name;  
 private final int age;  
  
 public Person(String name, int age) {  
 this.name = name;  
 this.age = age;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public int getAge() {  
 return age;  
 }  
}  
  
public class Task4 {  
 public static void main(String[] args) {  
 List<Person> people = new ArrayList<>();  
 people.add(new Person("Alice", 30));  
 people.add(new Person("Bob", 25));  
 people.add(new Person("Charlie", 35));  
  
 // Sort by age using a lambda expression  
 people.sort(Comparator.comparingInt(Person :: getAge));  
 // Print sorted list  
 for (Person person : people) {  
 System.out.println(person.getName() + " (" + person.getAge() + " years old)");  
 }  
 }  
}

**Output:**



**Task 5: Functional Interfaces**

Create a method that accepts functions as parameters using Predicate, Function, Consumer, and Supplier interfaces to operate on a Person object.

**Program**

*package* Assignments.Day19;  
  
*import* java.util.function.*Consumer*;  
*import* java.util.function.*Function*;  
*import* java.util.function.*Predicate*;  
*import* java.util.function.*Supplier*;  
  
  
  
*public class* Task5 {  
 *public static void* main(String[] args) {  
   
 Person person = *new* Person("Alice", 30);  
  
 *// Predicate: Check if the person is older than 25  
 Predicate*<Person> isOlderThan25 = p -> p.getAge() > 25;  
  
 *// Function: Convert a Person object to a formatted string  
 Function*<Person, String> personToString = p -> p.getName() + " (" + p.getAge() + " years old)";  
  
 *// Consumer: Print the person's details  
 Consumer*<Person> printPersonDetails = p -> System.***out***.println("Person details: " + p);  
  
 *// Supplier: Create a new Person object  
 Supplier*<Person> createNewPerson = () -> *new* Person("Bob", 22);  
  
 *// Example usage:  
 if* (isOlderThan25.test(person)) {  
 System.***out***.println(personToString.apply(person));  
 }  
  
 printPersonDetails.accept(person);  
  
 Person newPerson = createNewPerson.get();  
 System.***out***.println("New person created: " + newPerson);  
 }  
}

**Output:**

