Q1:

class Box<T> {

private T item;

public void setItem(T item) {

this.item = item;

}

public T getItem() {

return item;

}

}

public class Main {

public static void main(String[] args) {

Box<String> stringBox = new Box<>();

stringBox.setItem("Hello");

System.out.println(stringBox.getItem());

Box<Integer> intBox = new Box<>();

intBox.setItem(123);

System.out.println(intBox.getItem());

Box rawBox = new Box();

rawBox.setItem("Test");

System.out.println(rawBox.getItem());

}

}

**Question**: Find and explain the error or warning in the program. If there is no error, explain the output.

**Explanation:** There is no compilation error, but using a raw type Box without specifying a type parameter will generate an unchecked warning. The output will be "Hello", "123", and "Test".

**Mark**: 1 mark for no error, 1 mark for explaining the warning and output

**Q2:**

class Pair<K, V> {

private K key;

private V value;

public Pair(K key, V value) {

this.key = key;

this.value = value;

}

public K getKey() {

return key;

}

public V getValue() {

return value;

}

}

public class Main {

public static void main(String[] args) {

Pair<Integer, String> pair = new Pair<>(1, "One");

System.out.println("Key: " + pair.getKey() + ", Value: " + pair.getValue());

Pair<Double, Integer> invalidPair = new Pair<>(1.5, "Two");

System.out.println("Key: " + invalidPair.getKey() + ", Value: " + invalidPair.getValue());

}

}

**Question:** Find and explain the error or warning in the program. If there is no error, explain the output.

**Explanation:** There is a compilation error in the creation of invalidPair. The type parameters <Double, Integer> do not match the actual arguments provided (1.5, "Two").

**Mark:** 1 mark for error and 1 mark for explanation

**Q3:**

**import java.util.ArrayList;**

**class Container<T> {**

**private ArrayList<T> items = new ArrayList<>();**

**public void addItem(T item) {**

**items.add(item);**

**}**

**public T getItem(int index) {**

**return items.get(index);**

**}**

**}**

**public class Main {**

**public static void main(String[] args) {**

**Container<String> stringContainer = new Container<>();**

**stringContainer.addItem("Hello");**

**stringContainer.addItem("World");**

**System.out.println(stringContainer.getItem(0));**

**System.out.println(stringContainer.getItem(1));**

**Container<Integer> intContainer = new Container<>();**

**intContainer.addItem(1);**

**intContainer.addItem(2);**

**System.out.println(intContainer.getItem(0));**

**System.out.println(intContainer.getItem(1));**

**Container mixedContainer = new Container();**

**mixedContainer.addItem("String");**

**mixedContainer.addItem(100);**

**System.out.println(mixedContainer.getItem(0));**

**System.out.println(mixedContainer.getItem(1));**

**}**

**}**

**Question:** Find and explain the error or warning in the program. If there is no error, explain the output.

**Explanation:** The program will run with unchecked warnings because mixedContainer uses a raw type without specifying a generic type. This can lead to potential ClassCastException at runtime. The output will be "Hello", "World", "1", "2", "String", and "100".

**Mark:** 1 mark for no error, 1 mark for explaining the warning and output

**Q4:**

**class Printer<T extends Number> {**

**public void print(T value) {**

**System.out.println("Value: " + value);**

**}**

**}**

**public class Main {**

**public static void main(String[] args) {**

**Printer<Integer> intPrinter = new Printer<>();**

**intPrinter.print(100);**

**Printer<Double> doublePrinter = new Printer<>();**

**doublePrinter.print(10.5);**

**Printer<String> stringPrinter = new Printer<>();**

**stringPrinter.print("Hello");**

**}**

**}**

**Question:** Find and explain the error in the program. If there is no error, explain the output.

**Explanation:** There is a compilation error in the creation of stringPrinter. The Printer class is restricted to types that extend Number, so String is not a valid type argument.

**Mark:** 1 mark for error and 1 mark for explanation

Q5:

**import java.util.List;**

**class DataProcessor {**

**public static <T> void printList(List<T> list) {**

**for (T element : list) {**

**System.out.println(element);**

**}**

**}**

**}**

**public class Main {**

**public static void main(String[] args) {**

**List<Integer> intList = List.of(1, 2, 3, 4);**

**DataProcessor.printList(intList);**

**List<String> strList = List.of("One", "Two", "Three");**

**DataProcessor.printList(strList);**

**List rawList = List.of(1, "Two", 3.0);**

**DataProcessor.printList(rawList);**

**}**

**}**

**Question:** Find and explain the error or warning in the program. If there is no error, explain the output.

**Explanation:** The program will run successfully, but using a raw type List without specifying a type parameter will generate an unchecked warning. The output will be the elements of each list printed on separate lines.

Mark: 1 mark for stating error and 1 mark for explanation

Q6:

class Wrapper<T> {

private T value;

public Wrapper(T value) {

this.value = value;

}

public T getValue() {

return value;

}

}

public class Main {

public static void main(String[] args) {

Wrapper<String> stringWrapper = new Wrapper<>("Hello");

System.out.println(stringWrapper.getValue());

Wrapper<Double> doubleWrapper = new Wrapper<>(3.14);

System.out.println(doubleWrapper.getValue());

Wrapper intWrapper = new Wrapper(10);

System.out.println(intWrapper.getValue());

}

}

**Question:** Find and explain the error or warning in the program. If there is no error, explain the output.

**Explanation:** The program will compile and run, but using a raw type Wrapper without specifying a type parameter will generate an unchecked warning. The output will be "Hello", "3.14", and "10".

**Mark: 1 mark for no error, 1 mark for explanation about warning and output**

**Q7:**

**import java.util.ArrayList;**

**class GenericStack<T> {**

**private ArrayList<T> stack = new ArrayList<>();**

**public void push(T item) {**

**stack.add(item);**

**}**

**public T pop() {**

**if (!stack.isEmpty()) {**

**return stack.remove(stack.size() - 1);**

**}**

**return null;**

**}**

**}**

**public class Main {**

**public static void main(String[] args) {**

**GenericStack<String> stringStack = new GenericStack<>();**

**stringStack.push("A");**

**stringStack.push("B");**

**System.out.println(stringStack.pop());**

**System.out.println(stringStack.pop());**

**GenericStack<Integer> intStack = new GenericStack<>();**

**intStack.push(1);**

**intStack.push(2);**

**System.out.println(intStack.pop());**

**System.out.println(intStack.pop());**

**GenericStack rawStack = new GenericStack();**

**rawStack.push("C");**

**rawStack.push(3);**

**System.out.println(rawStack.pop());**

**System.out.println(rawStack.pop());**

**}**

**}**

**Question:** Find and explain the error or warning in the program. If there is no error, explain the output.

**Explanation:** The program will run successfully, but using a raw type GenericStack without specifying a type parameter will generate an unchecked warning. The output will be "B", "A", "2", "1", "3", and "C". Mixing types in the raw stack could lead to runtime errors.

**Mark**: 1 mark for no error, 1 mark for explanation about warning and output

**Q8:**

**class Pair<K, V> {**

**private K key;**

**private V value;**

**public Pair(K key, V value) {**

**this.key = key;**

**this.value = value;**

**}**

**public K getKey() {**

**return key;**

**}**

**public V getValue() {**

**return value;**

**}**

**public void display() {**

**System.out.println("Key: " + key + ", Value: " + value);**

**}**

**}**

**public class Main {**

**public static void main(String[] args) {**

**Pair<String, Integer> pair1 = new Pair<>("Age", 25);**

**pair1.display();**

**Pair<String, String> pair2 = new Pair<>("Name", "Alice");**

**pair2.display();**

**Pair<Integer, String> pair3 = new Pair<>(101, "Room");**

**pair3.display();**

**Pair mixedPair = new Pair("Key", 123.45);**

**mixedPair.display();**

**}**

**}**

**Question:** Find and explain the error or warning in the program. If there is no error, explain the output.

**Explanation:** The program will run, but using a raw type Pair without specifying a type parameter will generate an unchecked warning. The output will display each pair's key and value, with the mixed pair showing "Key: Key, Value: 123.45".

**Mark:** 1 mark for no error, 1 mark for explanation about warning and output

**Q9:**

**class Calculator<T extends Number> {**

**public double add(T num1, T num2) {**

**return num1.doubleValue() + num2.doubleValue();**

**}**

**}**

**public class Main {**

**public static void main(String[] args) {**

**Calculator<Integer> intCalculator = new Calculator<>();**

**System.out.println("Sum: " + intCalculator.add(10, 20));**

**Calculator<Double> doubleCalculator = new Calculator<>();**

**System.out.println("Sum: " + doubleCalculator.add(5.5, 2.5));**

**Calculator<Float> floatCalculator = new Calculator<>();**

**System.out.println("Sum: " + floatCalculator.add(3.0f, 4.5f));**

**Calculator<String> stringCalculator = new Calculator<>();**

**System.out.println("Sum: " + stringCalculator.add("10", "20"));**

**}**

**}**

**Question:** Find and explain the error or warning in the program. If there is no error, explain the output.

**Explanation:** There is a compilation error in the creation of stringCalculator. The Calculator class is restricted to types that extend Number, so String is not a valid type argument.

**Mark:** One mark for compilation error and 1 mark for explanation

**Q10:**

**import java.util.List;**

**class DataPrinter {**

**public static <T> void printArray(T[] array) {**

**for (T element : array) {**

**System.out.println(element);**

**}**

**}**

**}**

**public class Main {**

**public static void main(String[] args) {**

**Integer[] intArray = {1, 2, 3, 4};**

**DataPrinter.printArray(intArray);**

**String[] strArray = {"One", "Two", "Three"};**

**DataPrinter.printArray(strArray);**

**Double[] doubleArray = {1.1, 2.2, 3.3};**

**DataPrinter.printArray(doubleArray);**

**Object[] mixedArray = {"Hello", 123, 4.56};**

**DataPrinter.printArray(mixedArray);**

**}**

**}**

**Question:** Find and explain the error or warning in the program. If there is no error, explain the output.

**Explanation**: There is no error in the program. The output will be each element of the arrays printed on separate lines.

Mark: 1 mark for no error, 1 mark for explanation