Q1. Why is primitive type not allowed inside ArrayList but wrapper class is allowed? Explain with example.

Ans. Java Generics work only with **objects**, not with **primitive data types**. ArrayList<int> is invalid because int is **not an object**. Instead, we use **wrapper classes** like Integer for int. For ex -

```
import java.util.*;

public class Main {
   public static void main(String[] args) {
      // ArrayList<int> list = new ArrayList<>();  //  Not allowed
      ArrayList<Integer> list = new ArrayList<>();  //  Allowed
      list.add(10);
      list.add(20);
      System.out.println(list);
   }
}
```

Output :- [10, 20]

Q2. Write a Java program to convert a primitive double to a String without using autoboxing.

Ans.

```
public class Main {
   public static void main(String[] args) {
      double num = 45.67;
      String str = Double.toString(num);
      System.out.println("String value: " + str);
   }
}
```

Output :– String value: 45.67

Q3. What will happen if you try to add a primitive value to a collection without autoboxing? Show with code.

Ans. Without autoboxing, the compiler cannot convert primitive to an object automatically, and it will cause a compilation error. For ex -

```
import java.util.*;

public class Main {

   public static void main(String[] args) {

      ArrayList<Integer> list = new ArrayList<>();

      // list.add(5); // Works with autoboxing

      // Without autoboxing:

      Integer num = Integer.valueOf(5);

      list.add(num);

      System.out.println(list);

   }

}
```

Output :- [5]

Q4. Convert a String that contains a floating number to Double and print its integer value using intValue().

Ans.

```
public class Main {

   public static void main(String[] args) {

      String s = "45.89";

      Double d = Double.parseDouble(s);

      int intValue = d.intValue();

      System.out.println("Integer value: " + intValue);

   }

}
```

Output :- Integer value: 45

Q5. Compare two Integer objects using both == and equals() and explain the outputs.

Ans.

```
public class Main {
   public static void main(String[] args) {
      Integer a = 100;
      Integer b = 100;
      Integer c = 200;
      Integer d = 200;
      System.out.println(a == b);      // true (values between -128 to 127 cached)
      System.out.println(c == d);      // false (new objects)
      System.out.println(a.equals(b)); // true
      System.out.println(c.equals(d)); // true
   }
}
```

Output :-

```
true
false
true
true
```

Q6. Demonstrate NumberFormatException using Integer.parseInt() with invalid input.

Ans.

```
public class Main {
   public static void main(String[] args) {
      try {
         int num = Integer.parseInt("12AB"); // invalid number
      } catch (NumberFormatException e) {
```

```
            System.out.println("Exception caught: " + e);
        }
    }
}
```

Output :- Exception caught: java.lang.NumberFormatException: For input string: "12AB"

Q7. Write code to convert int → Integer → String → Integer and print each step.

Ans.

```
public class Main {
    public static void main(String[] args) {
        int num = 25;
        Integer obj = Integer.valueOf(num);  // int → Integer
        String str = obj.toString();         // Integer → String
        Integer finalObj = Integer.parseInt(str); // String → Integer
        System.out.println("int: " + num);
        System.out.println("Integer: " + obj);
        System.out.println("String: " + str);
        System.out.println("Back to Integer: " + finalObj);
    }
}
```

Output :-

int: 25

Integer: 25

String: 25

Back to Integer: 25

Q8. Create an ArrayList of Character type and check whether each character is a digit or alphabet.

Ans.

```
import java.util.*;

public class Main {
   public static void main(String[] args) {
      ArrayList<Character> list = new ArrayList<>();
      list.add('A');
      list.add('5');
      list.add('b');
      for (char ch : list) {
         if (Character.isDigit(ch))
            System.out.println(ch + " is a Digit");
         else if (Character.isLetter(ch))
            System.out.println(ch + " is an Alphabet");
         else
            System.out.println(ch + " is neither");
      }
   }
}
```

Output :–

```
A is an Alphabet
5 is a Digit
b is an Alphabet
```

Q9. Why can't a primitive type be used in Generics? Provide explanation and example.

Ans. Generics in Java work only with **objects**, and primitive types like int, char, double are **not objects**. Hence, you cannot use ArrayList<int>, but you can use ArrayList<Integer> (wrapper class). For ex-

```
import java.util.*;

public class Main {
   public static void main(String[] args) {
      // ArrayList<int> list = new ArrayList<>(); // Compile-time error
      ArrayList<Integer> list = new ArrayList<>(); // Correct
      list.add(10);
      System.out.println(list);
   }
}
```

Output :- [10]

Q10. Write a program where the user enters a numeric string, convert it into Integer, and print the square value.

Ans.

```
import java.util.*;

public class Main {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter a numeric string: ");
      String input = sc.nextLine();
      try {
         int num = Integer.parseInt(input);
         System.out.println("Square: " + (num * num));
      } catch (NumberFormatException e) {
         System.out.println("Invalid input! Please enter a numeric string.");
```

```
        }
    }
}
```

Output :–

Enter a numeric string: 12

Square: 144