1. Consider the following program fragment.

```
1  class A {
2    int x;
3
4    A(int x) {
5      this.x = x;
6    }
7
8    public A method() {
9      return new A(x);
10   }
11 }
12
13 class B extends A {
14   B(int x) {
15     super(x);
16   }
17
18   @Override
19   public B method() {
20     return new B(x);
21   }
22 }
```

   Does it compile? What happens if we switch the `method` definitions between class `A` and class `B` instead? Give reasons for your observations.

2. Consider a generic class `A<T>` with a type parameter `T` with a default constructor. Which of the following expressions are valid (*i.e.*, *with no compilation error*) ways of creating a new object of type `A`? We still consider the expression as valid if the Java compiler produces a warning.

   (a) `new A<int>();`

   (b) `new A<>();`

   (c) `new A();`

3. Compile and run the following program fragments and explain your observations.

   (a) Program A

```
1  import java.util.List;
2
3  class A {
4    void foo(List<Integer> integerList) {}
5    void foo(List<String> stringList) {}
6  }
```

(b) Program B

```
class B<T> {
  T x;
  static T y;
}
```

(c) Program B

```
class C<T> {
  static int b = 0;
  C() {
    this.b++;
  }
  public static void main(String[] args) {
    C<Integer> x = new C<>();
    C<String> y = new C<>();

    System.out.println(x.b);
    System.out.println(y.b);
  }
}
```