**Name:**

**Quiz#4 Generics**

String[] data = new String[1];

Object[] ptr;

1. Using the above explain why using Generics is beneficial <2 marks>

ptr = data;

ptr[0] = 15;//compiler cannot check if this isn’t allowed since ptr is an Object[] it assumes it’s fine

Using Generics type checking is done at COMPILE time. Arrays bypass compiler type checking and it must be done during runtime.

void copy(Object[] a, Object[] b){

System.array.copy(a,0,a.length,b,0,b.length);//assumes array b[] is large enough

}

1. Demonstrate how you can use Generics to improve the above copy() method and WHY it is better than the original method <2 marks>

<T> void copy(T[] a, copy(T[] b){

….

}

Since compiler can type check that BOTH arrays are the same type the runtime does not need to.

Interface Comparable<X>{

int compareTo(X x);

}

public static int min(int[]data){

int result = data[0];

for (int value : data){

if (result > value)

result = value;

}

return result;

}

1. Write the above function as a generic function. <4 marks>

pubic static <T extends Comparable<T>> T min(T[] data){

T result = data[0];

for(T value : data){

if (result.compareTo(value) > 0)

result = value;

}

return result;

}