AP Computer Science A
Parameter Passing Worksheet

Name ____
Date ____________________________ Period __________

Determine what the method does and prints, whether it can compile, and what runtime (execution) errors would be produced.

1) Local parameter name vs field variable name

```
private int fieldInt;

public void localVsFieldProblem() {
       int x = 5;    fieldInt = 123;
       System.out.println("Before localVsField: x = " + x + "  fieldInt = " + fieldInt);
       localVsField(x, fieldInt);
       System.out.println("After  localVsField: x = " + x + "  fieldInt = " + fieldInt);
}
public void localVsField(int x, int fieldInt) {
       x = 12;       fieldInt = 412;
       System.out.println("During localVsField: x = " + x + "  fieldInt = " + fieldInt);
}
```

2) Passing String (immutable), then manipulating in method

```
public void immutableParameterProblem() {
       String str = "hello";
       System.out.println("Before immutableParameter: str = " + str);
       immutableParameter(str);
       System.out.println("After  immutableParameter: str = " + str);
}
public void immutableParameter(String str) {
       str = "goodbye";
       System.out.println("During immutableParameter: str = " + str);
}
```

3) Passing mutable object, then manipulating in method

```
public void mutableParameterProblem() {
       java.awt.Point p = new java.awt.Point(10, 20);
       System.out.println("Before mutableParameter1: p = " + p);
       mutableParameter1(p);
       System.out.println("After  mutableParameter1: p = " + p);
       mutableParameter2(p);
       System.out.println("After  mutableParameter2: p = " + p);
}
public void mutableParameter1(java.awt.Point p) {
       p.setLocation(111, 222);
       System.out.println("During mutableParameter1: p = " + p);
}
public void mutableParameter2(java.awt.Point p) {
       p = new java.awt.Point(128, 128);
       System.out.println("During mutableParameter2: p = " + p);
}
```

4) Passing array, creating new array in method

```
public void arrayCreationProblem() {
       Double[] dblArr = new Double[] { 3.2, 5.1 };
       System.out.print("Before arrayCreation: dblArr = ");
       for (int a = 0; a < dblArr.length; a++) System.out.print(dblArr[a] + " ");
       System.out.println();
       arrayCreation(dblArr);
       System.out.print("After  arrayCreation: dblArr = ");
       for (int a = 0; a < dblArr.length; a++) System.out.print(dblArr[a] + " ");
}  // cont >>>>
```

```
public void arrayCreation(Double[] dblArr) {
       dblArr = new Double[] { 23.4, 333.0 };
       System.out.print("During arrayCreation: dblArr = ");
       for (int a = 0; a < dblArr.length; a++) System.out.print(dblArr[a] + " ");
       System.out.println();
}
```

5) Passing array with immutable objects, changing objects in method

```
public void arrayChangeProblem() {
       String[] strArr = new String[] { "hello", "goodbye" };
       System.out.print("Before arrayChange: strArr = ");
       for (int a = 0; a < strArr.length; a++) System.out.print(strArr[a] + " ");
       System.out.println();
       arrayChange(strArr);
       System.out.print("After  arrayChange: strArr = ");
       for (int a = 0; a < strArr.length; a++) System.out.print(strArr[a] + " ");
}
public void arrayChange(String[] strArr) {
       strArr[strArr.length - 1] = "vacation";
       System.out.print("During arrayChange: strArr = ");
       for (int a = 0; a < strArr.length; a++) System.out.print(strArr[a] + " ");
       System.out.println();
}
```

6) Passing primitive, changing primitive in method

```
public void primitiveChangeProblem() {
       int a = 5;   boolean b = false;
       System.out.println("Before primitiveChange: a = " + a + "  b = " + b);
       primitiveChange(a, b);
       System.out.println("After  primitiveChange: a = " + a + "  b = " + b);
}
public void primitiveChange(int a, boolean b) {
       a = 321;
       b = true;
       System.out.println("During primitiveChange: a = " + a + "  b = " + b);
}
```

7) Working with static vs non-static (instance-based) fields, parameters, and methods

```
private int instanceInt;
private static int staticInt;

public void staticVsNonstaticProblems() {
       staticInt = 10;     instanceInt = 15;
       staticMethod(staticInt, instanceInt);
       nonstaticMethod(staticInt, instanceInt);
}
// STATIC method
public static void staticMethod(int si, int nsi) {
       staticInt = si + nsi;      // ???? ERROR - Reason:

       instanceInt = si + nsi;    // ???? ERROR - Reason:

       nonstaticMethod(si, nsi);  // ???? ERROR - Reason:
}
// Instance-based method (NONSTATIC)
public void nonstaticMethod(int si, int nsi) {
       staticInt = si + nsi;      // ???? ERROR - Reason:

       instanceInt = si + nsi;    // ???? ERROR - Reason:
}
```