

1. Matching [10 pts]

Choose the **best** word for each statement.

1. _____ The name of the java compiler.
 2. _____ The java compiler converts the java source into this.
 3. _____ The file type of Java byte code.
 4. _____ A set of objects with the same behaviour.
 5. _____ Keyword used to declare constants in Java.
- A. Object Construction
 - B. final
 - C. Byte Code
 - D. Space Tracker
 - E. java
 - F. javac
 - G. Public method
 - H. Encapsulation
 - I. Private method
 - J. class
 - K. Photon
 - L. Feedback Control

2. Java Data Types [10 pts]

Circle all of the words that **are** Java primitive types.

for

String

int

triple

do

polymorphism

Object

double

short

byte

long

wide

boolean

java

char

3. Declaration and Initialization [10 pts]

Indicate whether or not the following statements are syntactically correct, ie they will compile correctly. If the statement is correct, write **OK** and proceed to the next part; if the statement is incorrect, write **ERROR** and re-write the statement so the problem is fixed. If you decide to write OK and make a fix to a part, that part will be marked entirely wrong.

2 (a) `int retVal = new int(9);`

2 (b) `String int = "Wicked Awesome";`

2 (c) `double d=9.0d;`

2 (d) `float a;`

2 (e) `String temp= new String();`

4. **Compilation and Execution [10 pts]**

- 5 (a) What command do you type at the command prompt to compile all of the java files in the directory?
- 5 (b) What command do you type at the command prompt to run the java class **MyClass**?

5. Loss of Precision and Casting [10 pts]

Given the following statements determine where a cast is needed. If no casting is needed, write **OK**; if casting is required because of possible loss of precision write **ERROR** and change the code so it compiles. No extraneous casts should be made: that means if it will compile without casting, then you shouldn't add a cast just in case.

- 2 (a) `double b = 9.19;`
`short a = (short) b;`
`int c = b;`
- 2 (b) `int trouble = 45;`
`char gobble = trouble;`
- 2 (c) `float myFloat = 32;`
- 2 (d) `int partd = 2343;`
`short partd2 = partd;`
- 2 (e) `double lastone = 32.0f;`

6. Arrays [10 pts]

Write the method **public double average(double[] a)** which returns the average of all the values inside the array passed in. You can assume that the array is of non-zero length.

```
public double average(double[] a) {
```

```
}
```

7. Iteration [10 pts]

Write the method `public int evens()` which returns the count of **even** numbers between 431 and 500124. 431 and 500124 should also be considered.. You **must use iteration** to solve this problem. If you do not use iteration, or if you use any recursion, you will receive no credit.

```
public int evens() {
```

```
}
```

8. Accessors and Mutators [10 pts]

Given the following class framework, write accessors and mutators for all the variables:

```
public class p8{  
  
    private String cool;  
    private int difficulty;  
    private double length;
```

```
} //end class p8
```

9. Loops [10 pts]

Convert the following **for loop** into a **while loop**.

```
for(int i=3; i <= 45; i++){  
    i+=2;  
    System.out.println(i);  
}
```

10. **Tracing [10 pts]**

Given the following class, write the output when the class is run.

```
public class TraceA{

    public String printMe(){
        return "Wow";
    }//end printMe()

    public static void main(String[] argv){
        boolean a = true;
        boolean b = true;
        TraceA c = new TraceA();
        int j = 4;
        if( a && b ){
            System.out.println("Solo");
            a=false;
        }
        if( a || b ){
            System.out.println("U2");
        }
        j++;
        do{
            j--;
            System.out.println("In here");
        }while(j>3);
        System.out.println(j);
        c.printMe();
    }//end main(String[])
} //end class TraceA
```
