ICS 4M0 — Semester 2, 2008 - 2009

Grade 12 Computer and Information Science
Unit 0 – Major Test

Mr. N. Nolfi
Victim: Mr. Solutions Super work Mr. S.!!

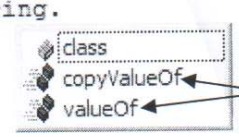
KU	APP	TIPS	COM
10/10	10/10	15/15	14/14

1. Identify the indicated parts in each Java code snippet. **(6 KU)**

(a)
```
String name=new String("Tahmoor");
```
- Name of: class
- Name of: object
- Call of: constructor method

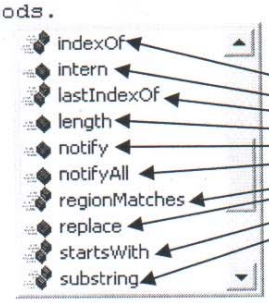
(b)
```
double averageEarthSunDistance=1.495978706e11;
```
- Name of: primitive data type
- Name of: variable
- initial value of: variable

(c)
```
distance = velocity/time
```
- assignment statement
- expression

(d)
```
String.
```


class or static methods

(e)
```
String favouriteFoods=new String("Fruits and Vegetables");

favouriteFoods.
```


instance methods

2. For the following Java code "snippet," complete a memory map and state the problem that is solved. **(5 APP, 3 COM)**

```
int bigMac=0;
String c=new String(), s=new String("01234"), a=new String("98054213");
for (int i=0; i<a.length(); i++){
   c=a.substring(i,i+1);
   if (s.indexOf(c)>=0)
      bigMac ++;
}
```

Before Loop | After Loop

i	c	bigMac
—	""	0
0	"9"	0
1	"8"	0
2	"0"	1
3	"5"	1
4	"4"	2
5	"2"	3
6	"1"	4
7	"3"	5
—	"3"	5

By the time the loop has finished executing, the variable "bigMac" stores

the number of digits, from 0 to 4 inclusive, found in the string stored in the string object "a"

3. The code given below is supposed to find the smallest integer, other than 1, that divides into "num." Unfortunately, the code was written by a very sloppy programmer who made several syntax, logic and indentation errors. Circle each error and write corrected code in the provided space. **(5 APP, 5 TIPS)**

Sloppy Code (Circle all errors)	*Corrected Code*

Two errors here
`int num =`
`  Integer.parseInt(edit1.getText)`
`int smallestDiv = num`
`for (int x=2, x >= num/2, ++x);{`
`if (num % x - 0);`
`smallestDiv = x`
`return; //exit loop prematurely`
space
`label1.setText("Smallest Divisor of "+`
`  num+"other than 1 is "+smallestDiv);`
`}//end of for`
	`int num = Integer.parseInt(edit1.getText());`
	`int smallestDiv = num;`
	`for (int x=2; x<=num/2; ++x) {`
	`  if (num % x == 0){`
	`  smallestDiv = x;`
	`  break; //exit loop prematurely`
	`  }//end of if`
	`}//end of for`
	`label1.setText("Smallest Divisor of "+`
	`  num+" other than 1 is "`
	`  +smallestDiv);`

4. *Overloading* is an important principle of object-oriented programming.

(a) Explain what is meant by overloading. **(2 KU, 2 COM)**

A method is said to be overloaded if there are two or more methods having exactly the same name. This is only allowed if the various versions of the method have parameters that differ in either number or type. That is, each version must accept at least one parameter whose type is different from all the rest or it must accept a number of parameters different from all the others.

(b) What is the main advantage of overloading? Explain. **(2 KU, 2 COM)**

Overloading makes it very easy and convenient for a particular method to handle many different types of data. For example, there are 11 different versions of the String constructor method, each of which accepts either a different kind of data or a different number of parameters. Nonetheless, each version of the method accomplishes the same basic task of initializing a string object.

5. An integer is *prime* if its only proper divisor is *one*. For example, 7 is prime because its only proper divisor is 1. However, 8 is not prime because its proper divisors are 1, 2 and 4.

(a) Explain the steps that a *computer* would need to execute to determine whether a given integer is prime. **(3 TIPS, 2 COM)**

- starting at 2, calculate the remainder obtained when the given number is divided by 2
- repeat this with 3, 4, 5, ... until half of the given # is reached
- if at any point the remainder is found to be zero, then the given # is not prime → otherwise, it is prime

(b) Write a Java program segment that determines whether a given integer is prime. Your code should include variable declarations. **(7 TIPS, 5 COM)**

```
int number = Integer.parseInt(editNumber.getText());
boolean prime = true;
for (i=2; i<=number/2; i++) {
    if (number % i == 0){
        prime = false;
        break; //exit loop prematurely because a divisor has been found
    } //end of if
} //end of for
if (prime) //This is equivalent to "if (prime==true)"
    labelPrime.setText(number + " is prime");
else
    labelPrime.setText(number + " is not prime");
```

if i is greater than half the number it cannot divide into the number.

This must be placed after the loop. First, the loop searches for proper divisors of the number. Once the search is complete, the decision of whether the number is prime can be made.