Tutorial letter 103/1/2018

Interactive Programming

ICT2612

Semesters 1

School of Computing

IMPORTANT INFORMATION:

This tutorial letter contains important information about your assignment 2.

DUE DATE: 16 APRIL 2018

UNIQUE CODE: 700302

INSTRUCTIONS

Work through all the questions and select the correct answer. When you are done, logon to myUNISA, select assignment 2 and complete and submit the online MCQ.

Study the incomplete code in CODE SECTION 1 and answer question 1 that follows.

CODE SECTION 1:

```
double axe1 = 15;
double axe2 = 25;
//--(i)--
```

Question 1 **(1)**

Indicate which of the following statements for the missing code in (i) will NOT render and error message.

(1) `double total = axe1 + axe2`

(2) `float total = axe1 + axe2;`

(3) `int total = axe1 + axe2;`

(4) `double total = axe1 + axe2;`

(5) None of the above

`Study the code in CODE SECTION 2 and answer question 2 that follows.

CODE SECTION 2:

```
double myTotal = 456.7;
int myIntTotal = (int)myTotal;
```

Question 2 **(1)**

Study the code in CODE SECTION 2 and indicate what the value of `myIntTotal` will be.

(1) Error message.

(2) 456

(3) 457

(4) 4567

(5) None of the above

Study the code in CODE SECTION 3 and answer question 3 that follows.

CODE SECTION 3:

```
String w = "1", x = "2", y = "3";
char z = '4';
String mess = w + x + y + z;
```

Question 3 **(1)**

Indicate what the value of `mess` will be.

(1) Error message. You cannot use the '+' to concatenate `String` and `char` values.

(2) 123 4

(3) 1234

(4) 9

(5) None of the above

Question 4 **(1)**

Indicate which one of the following declarations will render an error?

(1) long value = 456_789_101;

(2) int value = 22.0;

(3) int value = ' b';

(4) float value = 1.23f;

(5) boolean value = true;

Question 5 **(1)**

The data type ____ can store only a single Unicode character, e.g. 'B' or '&'.

(1) double

(2) byte

(3) char

(4) boolean

(5) float

Question 6 **(1)**

Which of the following is an INVALID variable name in Java?

(1) `String Jav = "Java";`

`Reason: A variable name cannot start with a capital letter.`

(2) `String jav = "Java";`

`Reason: "jav" is a reserved keyword in Java.`

(3) `String 3jav = "Java";`

`Reason: A variable name cannot start with a number.`

(4) `String _jav = "Java";`

`Reason: A variable name cannot start with an underscore (_).`

(5) `String jav$ = "Java";`

`Reason: The dollar sign ($) can only be used at the start of the variable name, e.g. $ jav.`

Study the code in CODE SECTION 4 and answer question 7 that follows.

CODE SECTION 4:

```
int res1 = 6;

++res1;

int res2 = res1--;
```

Question 7 **(1)**

Indicate what the values of `res1` and `res2` will be:

(1) `res1: 6  res2: 5`

(2) `res1: 6  res2: 6`

(3) `res1: 6  res2: 7`

(4) `res1: 7  res2: 6`

(5) `res1: 7  res2: 7`

Study the code in CODE SECTION 5 and answer question 8 that follows.

CODE SECTION 5:

Below is a formula to calculate the `Result` of three variables `T1`, `T2` and `T3` multiplied by R:

$$\text{Result} = \frac{T1+T2+T3}{3} \text{ x } R$$

The programmer created the following Java code to solve the above:

```
double Result;
double T1=2, T2=3, T3=4, R=5;
Result = calcResult(T1, T2, T3, R);
```

Question 8 **(1)**

Indicate which one of the following Java code represents the correct code for the method `calcResult()` that will solve the formula to calculate the `Result`.

(1)
```
private static double calcResult (double T1, double T2, double T3,
double R) {
 //calculate the result of the formula
   double answer;
   answer = (T1 + T2 + T3) / 3 x R;
  return (answer);
}
```

(2)
```
private static double calcResult (double T1, double T2, double T3,
double R) {
 //calculate the result of the formula
   double answer;
   answer = (T1 + T2 + T3)/ 3 * R;
  return (answer);
}
```

(3)
```
private static double calcResult (int T1, int T2, int T3, double R) {
 //calculate the area of a trapezoid
   double answer;
   answer = (T1 + T2 + T3) div 3 * R;
  return (answer);
}
```

(4)
```
private static double calcResult (int T1, int T2, int T3, double R) {
 //calculate the result of the formula
   double answer;
   answer = T1 + T2 + T3/ 3 * R;
  return (answer);
}
```

(5)
```
private static double calcResult (int T1, int T2, int T3, double R) {
 //calculate the area of a trapezoid
   double answer;
   answer = (T1 + T2 + T3) div 3 * R;
  return (answer);
}
```

Study the code in CODE SECTION 6 and answer question 9 that follows.

CODE SECTION 6:

```
String canBorrow = "No" ;
String videoNr = "1-345-6789";
String status = "1";
boolean available = status == "1";
boolean truth = (videoNr.substring(0,1).equals("1") && available);
if (truth) canBorrow = "Yes";
```

Question 9 **(1)**

Study the code in code section 6 and indicate what the values of the variables `available`, `truth` and `canBorrow` will be after execution of the code：

(1)
```
available: false
truth: false
canBorrow: No
```

(2)
```
available: true
truth: false
canBorrow: No
```

(3)
```
available: true
truth: false
canBorrow: No
```

(4)
```
available: true
truth: true
canBorrow: Yes
```

(5)
```
available: true
truth: false
canBorrow: Yes
```

Study SCENARIO (ICT1234) and incomplete code in CODE SECTION 7 and answer questions 10 to 14 that follows.

SCENARIO (ICT1234)

A student enrolling for the module ICT1234 has a total of seven (7) tests (TS) to complete. The YearMark is the average of the five (5) highest test marks. For example, if the test marks are 60,65,70,64,80,55,65 then only the marks 64,65,65,70 and 80 are taken into consideration.

Furthermore, the student submits a portfolio (PortFolio) and writes a written exam (ExamMark).

The FinalMark = YearMark x 30% + PortFolio x 50% + ExamMark x 20%

A student passes the module if the FinalMark is 50% or above AND the

ExamMark is 40% or above. If any of these two are not met, then the student fails the module ICT1234.

CODE SECTION 7:

```
//Declare the values
  double[] TS = {65.0,93.0,76.0,65.0,39.0,65.0,68.0};
  double YearMark = 0.0;
  double TotYearMark = 0.0;
  double PortFolio = 51.7;
  double ExamMark = 67.0;
  double FinalMark = 0.0;
  String Message = null;

// (i) Sort the array (TS)

// (ii) Select the top 5 values and determine the
//        sum thereof(TotYearMark)

//(iii) Calculate the YearMark

//(iv) Call method FM to calculate the FinalMark

//(v) Determine a Pass or Fail (Message)

//methods
private static double FM(double ym1, double pfm2, double
em3) {
// Calculates the FinalMark;
   double mark = ym1 * .3 + pfm2 * .5 + em3 * .2;
   return mark;
}//FM
```

Question 10 **(1)**

(i): Indicate which of the following code will correctly sort the array `TS` in ascending order.

(1)
```
Arrays.sort(TS);
```

(2)
```
Arrays.sort.asc(TS);
```

(3)
```
TS.sort;
```

(4)
```
double temp = 0.0;
for (int i = 0; i < TS.length-1; i++)
   for (int j = i+1; j < TS.length; j++){
       if (TS[i] < TS[j]){
         temp = TS[i];
         TS[i]= TS[j];
         TS[j] = temp;
        }
   }
```

(5) None of the above

Question 11 **(1)**

(ii): Indicate which of the following will select the sum of the top five test marks values to determine the year mark (`TotYearMark`).

(1)
```
for (int i = 1; i < TS.length-1; i++){
  TotYearMark = TS[i]++;
}
```

(2)
```
for (int i = 2; i < TS.length; i++){
  TotYearMark = ++TS[i];
}
```

(3)
```
for (int i = 1; i < TS.length-1; i++){
  TotYearMark = TotYearMark + TS[i];
}
```

(4)
```
for (int i = 2; i < TS.length; i++){
  TotYearMark = TotYearMark + TS[i];
}
```

(5) None of the above

Question 12 **(1)**

(iii): Indicate which of the following code will correctly determine the `YearMark`. The `YearMark` is the average of the top 5 test marks (`TotYearMark`).

(1)
```
YearMark = TotYearMark.average();
```

(2)
```
YearMark = TotYearMark / 5;
```

(3)
```
YearMark = Arrays.avg(TS);
```

(4)
```
YearMark = Arrays.total(TS) / 5;
```

(5) None of the above.

Question 13 **(1)**

(iv): Indicate which of the following code will correctly call the method `FM` that calculates the final year mark.

(1)
```
FM (YearMark, PortFolio, ExamMark);
```

(2)
```
FinalMark = FM (TS);
```

(3)
```
FinalMark = FM (YearMark, PortFolio, ExamMark);
```

(4)
```
FinalMark = FM (ym1,pf2,em3);
```

(5) None of the above

Question 14 **(1)**

(v): Indicate which of the following code will correctly determine whether the `FinalMark` is a "Pass" or "Fail".

(1)
```
Message = "Fail";
if ((FinalMark >= 50) && (ExamMark >= 40))
  Message = "Pass";
```

(2)
```
if ((FinalMark >= 50) || (ExamMark >= 40))
  Message = "Pass";
else
  Message = "Fail";
```

(3)
```
Message = "Fail";
if ((FinalMark >= 50) AND (ExamMark >= 40))
  Message = "Pass";
else
  Message = "Fail";
```

(4)
```
Message = "Fail";
if ((FinalMark >= 50) == (ExamMark >= 40))
  Message = "Pass";
else
  Message = "Fail";
```

(5) None of the above

Study the code in CODE SECTION 8 and answer question 15 that follows.

CODE SECTION 8:

```
int a = -1;  int b = 6;
String msg = "";
```

```
while (a <5) {
  b = --b;  a = a +1;
}//while
msg = a+" "+ b;
```

Question 15 **(1)**

Indicate what the value of `msg` will be.

(1) 5 0

(2) 4 1

(3) 5 -1

(4) 4 0

(5) The code will go into an endless loop.

Study the code in CODE SECTION 9 and answer question 16 that follows.

CODE SECTION 9:

```
String arrow="";
char input = 'l';

     if (input == 'w') {arrow = "up";}
else if (input == 'a') {arrow = "left";}
else if (input == 'd') {arrow = "right";}
else if (input == 'x') {arrow = "down";}
else arrow = "error";
```

Question 16 **(1)**

Indicate which of the following code can replace the code and still get the same results.

(1) `switch(input){`

```
 case 'w' : arrow = "up";
 case 'a' : arrow = "left";
 case 'd' : arrow = "right";
 case 'x' : arrow = "down";
 default: arrow = "error"; break;
}
```

(2)
```
switch(arrow){
 case 'w' : arrow = "up"; break;
 case 'a' : arrow = "left"; break;
 case 'd' : arrow = "right"; break;
 case 'x' : arrow = "down"; break;
 default: arrow = "error";
}
```

(3)
```
case(input){
 switch
    'w' : arrow = "up"; break;
    'a' : arrow = "left"; break;
    'd' : arrow = "right"; break;
    'x' : arrow = "down"; break;
 default: arrow = "error";
}
```

(4)
```
switch(input){
 case 'w' : arrow = "up"; break;
 case 'a' : arrow = "left"; break;
 case 'd' : arrow = "right"; break;
 case 'x' : arrow = "down"; break;
 default: arrow = "error";
}
```

(5) None of the above.

Study the code in CODE SECTION 10 and answer question 17 that follows.

CODE SECTION 10

```
int grade = 85;
String redoStudent="";
boolean madeFailRoll = false;

//(i) Statement to determine if the grade is 49% or below
if (grade <= 49) {madeFailRoll = true;}

if(madeFailRoll) redoStudent = "You made the Redo Roll";
```

Question 17 **(1)**

Indicate which of the following statements can replace the code in (i) and still get the same results.

(1)
```
madeFailRoll = grade <= 49;
```

(2)
```
madeFailRoll == grade <= 49;
```

(3)
```
switch(grade){
  case grade <= 49 : madeFailRoll = true;
}
```

(4)
```
switch(grade){
  case <= 49 : madeFailRoll = true;
}
```

(5) None of the above

Study the code in CODE SECTION 11 and CODE SECTION 12 and answer

questions 18 to 24 that follows.

CODE SECTION 11

```
Bike newBike = new Bike();
Bike newBike1 = new Bike ("Bling","Unleaded Petrol");
Bike newBike2 = new Bike ("Sophisticated", "Petrol","Auto");
Bike newBike3 = new Bike ("Dangerous","Petrol","Manual");
newBike1.setBasicSpecs();
newBike2.setBasicSpecs();
newBike3.setBasicSpecs();

int numberOfBikes = newBike.countBikes;

String basicSpecs1 = newBike1.getBasicSpecs();
String moreSpecs1 = newBike1.getMoreSpecs();

newBike2.topSpeed = "200km/h";
String basicSpecs2 = newBike2.getBasicSpecs();
String moreSpecs2 = newBike2.getMoreSpecs();

String basicSpecs3 = newBike3.getBasicSpecs();
String moreSpecs3 = newBike3.getMoreSpecs();

String drivenWheels = newBike3.drivenWheels;

//(i)
if (drivenWheels.equals("TWD"))
  {drivenWheels = "Two Wheel Drive";}
else
   {drivenWheels = "Four Wheel Drive";}

-----------------------------------------------------
```

CODE SECTION 12

```
public class Bike {

 static public int countBikes;

 public String enjinCapacity;
 public String steering;
 public String aircondition;
 public String seatingCapacity;
 public String drivenWheels;
 public String topSpeed;
 private String model;
 private String serviceIntervals;
 private String fuelSystem;
 private String emission;
 private String transmission;
 private String seatTrim;
 private String startingSystem;

//constructors

public Bike(){
//empty constructor
}

public Bike(String model, String fs) {
// Specific Specs

  countBikes++;

  this.model = model;
  this.fuelSystem = fs;
  emission = "Canadian 1";
```

```
  serviceIntervals = "5000";
  transmission = "Manual 4 Speed";
  seatTrim = "Synthetic Leather";
  startingSystem = "Rotary";
  topSpeed = "140km/h";
}

public Bike(String model, String fs, String tm) {
// Specific specs
  this.model = model;
  this.fuelSystem = fs;
  this.transmission = tm;
  emission = "Euro 2";
  serviceIntervals = "10000";
  seatTrim = "Leather";
  startingSystem = "Kick Start";
  topSpeed = "200km/h";
}

public void setBasicSpecs() {
// Basic Specs
  enjinCapacity = "1.5";
  steering = "electric";
  aircondition = "yes";
  seatingCapacity = "2";
  String fuelTankSize = "30";
  drivenWheels = "TWD";
}//setBasicSpecs

public String getBasicSpecs() {
// get Specific specs and return to calling program

 String msg = null;
```

```
 msg = this.model + " " + this.enjinCapacity + " " +
        this.fuelSystem + " " + this.serviceIntervals + " "
        + this.transmission + " " + this.seatTrim;
 return msg;

}//getBasicSpecs
public String getMoreSpecs() {
// get more technical specs and return to calling program
 return this.model + " " + this.emission + " " +
        this.startingSystem + " " + this.topSpeed;
}//getMoreSpecs
}
```

Question 18 **(1)**

`msg` in the method `getBasicSpecs()` in the class `Bike` is an example of ….

(1) a local variable

(2) an instance variable

(3) a class variable

(4) a static variable

(5) None of the above.

(1) a local variable

Question 19 **(1)**

`enjinCapacity` and `startingSystem` in the class `Bike` are examples of …

(1) local variables

(2) instance variables

(3) class variables

(4) static variables

(5) None of the above.

Question 20 **(1)**

Indicate what the value of `basicSpecs1` will be.

(1) `Bling Unleaded Petrol 5000 Manual 4 Speed Synthetic Leather`

(2) `Bling null Unleaded Petrol 5000 Manual 4 Speed Synthetic Leather`

(3) `Bling 1.5 Unleaded Petrol 5000 Manual 4 Speed Synthetic Leather`

(4) Error message. The variable `enjinCapacity` is a local variable in the method `setBasicSpecs`.

(5) None of the above.

Question 21 **(1)**

Indicate what the value of `moreSpecs2` will be

(1) Error message. You cannot change the value of `newBike2.topSpeed` from the main program.

(2) `Sophisticated Euro 2 Kick Start null`

(3) `Sophisticated Euro 2 Kick Start 200km/h`

(4) `Sophisticated Euro 2 Kick Start 140km/h`

(5) None of the above.

Question 22 **(1)**

The user enters the following code in `Main` but receives an error message:

```
newBike3.model = "Blitz ";
```

This error message is because ….

(1) the field `newBike3.model` does not exist.

(2) the field `Bike.model` does not exist.

(3) the code should be `Bike.model = "Blitz";`

(4) the field `Bike.model` is not visible in the main program.

(5) None of the above

Question 23 **(1)**

The if-then-else statement in the last section of the code (i) can be replaced with the following ternary operator…

(1)
```
drivenWheels =
((newBike3.drivenWheels=="TWD") ?
  {"Two Wheel Drive"} : {"Four Wheel Drive"});
```

(2)
```
drivenWheels ==
((newBike3.drivenWheels=="TWD") ?
  {"Two Wheel Drive"} : {"Four Wheel Drive"});
```

(3)
```
drivenWheels.equals
((newBike3.drivenWheels=="TWD") ?
  {"Two Wheel Drive"} : {"Four Wheel Drive"});
```

(4)
```
drivenWheels =
((newBike3.drivenWheels=="TWD") ?
  "Two Wheel Drive" : "Four Wheel Drive");
```

(5) None of the above.

Question 24 **(1)**

The value of `numberOfBikes` in the `Main` program will be …

(1) 0.

You cannot access the static value `countBikes` in the class Car from within the main program.

(2) 0

The statement `countBikes++` in the constructor `Bike(String, String)` will render a syntax error as you can not update the value of a static variable.

(3) 1

(4) 3

(5) None of the above.

Study the incomplete code in SECTION 13 and answer questions 25 and 26 that follow.

CODE SECTION 13

```
String mess="";

ArrayList<Integer> entity = new ArrayList<>();

//(i)


int count = entity.size();


//(ii)
```

Question 25 **(1)**

(i) Which of the following code can be used to allocate values to the `ArrayList entity`?

(1)
```
entity[0]=10;

entity[1]=15;

entity [2]=20;
```

(2)
```
entity.add[0]=10;

entity.add[1]=15;

entity.add[2]=20;
```

(3)
```
entity.add[10];

entity.add[15];

entity.add[20];
```

(4)
```
entity.add(10);

entity.add(15);

entity.add(20);
```

(5) None of the above. The size of the arraylist is not defined.

Question 26 **(1)**

(ii) Which of the following code can be used to loop through the elements of the arraylist?

(1)
```
for (int i = 0; i < count; i++) {
  int value = entity [i];
  mess = mess + " " + value;
}
```

(2)
```
for (int i = 0; i < count; i++) {
  int value = entity.get(i);
  mess = mess + " " + value;
}
```

(3)
```
for (int i = 0; i < count; i++) {
  int value = entity.get[i];
  mess = mess + " " + value;}
```

(4)
```
for (int i = 0; i < count; i++) {
  int value = entity[i].get();
  mess = mess + " " + value;
}
```

(5) None of the above.

Study the incomplete code in SECTIONS 14, 15 and 16 answer questions 27 and 28 that follow.

CODE SECTION 14

```
String[]                    TownNames                    =
{"Pretoria","Johannesburg","Bloemfontein”,
              "eMbalentli","Queenstown","Cape Town”,
                 "Durban","Port Shepstone",
                 ……};
```

```
int position = findDuplicates(TownNames);

String mess = "";

if (position < 0) {mess = "there is no duplicate";}
else  {mess = "the duplicate is " + TownNames[position] +
             " at positon " + position;}

TownNames = removeDuplicate(TownNames,position) ;
```

CODE SECTION 15

```
private static int findDuplicates(String[] Names) {
 int position = -1;

//--i---
 return position;

}//findDuplicates
```

CODE SECTION 16

```
private static String[] removeDuplicate
                            (String[] Names, int position) {

 String[] newList = new String[Names.length-1];
 int duplicate = position;

  //--ii--
   Names = newList;
  return Names;

}//removeDuplicate
```

Question 27 **(1)**

The user creates an array of all the names of primary schools in the Gauteng area (refer to CODE SECTION 14).

The name for every town must be entered only once. The array is quite large.

After a while he is not sure as to whether he has entered a town twice.

Indicate which of the following methods (CODE SECTION 15 – (i)) will correctly determine whether the name of a town is already entered in the array and return the position thereof in the array.

(1)
```
String duplicate = null;
 for (int i = 0; i < Names.length; i++){
  for (int j = i + 1; j < Names.length; j++){
    if (Names[i]==(Names[j+1])){
         duplicate = Names[j+1];
         position = j;
    }//if
   }//j
  }//i
```

(2)
```
String duplicate = null;
 for (int i = 0; i < Names.length; i++){
  for (int j = i + 1; j < Names.length; j++){
    if (Names[i]<(Names[j+1])){
         duplicate = Names[j+1];
         position = j;
    }//if
   }//j
  }//i
```

(3)
```
String duplicate = null;
 for (int i = 0; i < Names.length; i++){
  for (int j = i + 1; j < Names.length; j++){
    if (Names[i].equals(Names[j])){
```

```
            duplicate = Names[j];
            position = j;
          }//if
         }//j
        }//i
```

(4)
```
String duplicate = null;
 for (int i = 0; i < Names.length; i++){
  for (int j = i + 1; j < Names.length-1; j++){
    if (Names[i].equals(Names[j+1])){
            duplicate = Names[j];
            position = j;
    }//if
   }//j
  }//i
```

(5) None of the above.

Question 28 **(1)**

Indicate which of the following methods (CODE SECTION 16 – (ii)) will correctly remove the duplicate element from the array and return an array without the duplicate value.

(1)
```
for (int i = 0; i<position; i++){
  newList[i] = Names[i];
}
int j = position;
for (int i = position+1; i < Names.length; i++){
            newList[j] = Names[i];
            j++;
}
```

(2)
```
for (int i = 0; i<position; i++){
  newList[i] = Names[i];
}
int j = position+1;
```

```
for (int i = position; i < Names.length; i++){
        newList[j] = Names[i];
        j++;
}
```

(3)
```
for (int i = 0; i<position; i++){
  newList[i] = Names[i];
}

int j = position+1;
for (int i = position; i < Names.length; i++){
        newList[j+1] = Names[i];
        j++;
}
```

(4)
```
for (int i = 0; i<position; i++){
  newList[i] = Names[i];
}

int j = position+1;
for (int i = position; i < Names.length; i++){
        newList[i] = Names[i];
        j++;
}
```

(5) None of the above.

Question 29 **(1)**

Study the code below and indicate which of the following methods **CANNOT** be used to create the method `calc()` ?

```
boolean pass;
int examMark = 50;
```

```
int yearMark = 40;


fail = calc(examMark,yearMark);
```

(1)
```
private static boolean calc
                    (int examMark, int yearMark) {

    double finalMark = (examMark + yearMark) / 2;
    boolean pass=false;
    if (finalMark <= 49) {pass = false;}
    else {pass = true;}
    return pass;
  }
```

(2)
```
private static boolean calc
                    (int examMark, int yearMark) {

    double finalMark = (examMark + yearMark) / 2;
    boolean pass=false;
    if (finalMark > 49) {pass = true;}
    return pass;
  }
```

(3)
```
private static boolean calc
                    (double examMark, double yearMark){
    double finalMark = (examMark + yearMark) / 2;
    boolean pass=false;
    if (finalMark <= 49) {pass = false;}
     {pass = true;}
    return pass;
  }
```

(4)
```
private static boolean calc
                (int examMark, int yearMark) {
  double finalMark = (examMark + yearMark) / 2;
  boolean pass = (finalMark > 49) ? true : false;
  return pass;
}
```

(5)
```
private static boolean calc
                    (double examMark, double yearMark){
  double finalMark = (examMark + yearMark) / 2;
  boolean pass=true;
  if (finalMark <= 49) {pass = false;}
  return pass;
}
```

Question 30 **(1)**

Which one of the following lines of code is an INCORRECT array declaration?

```
int myArr1[] = new int[5];           //line 1
int [] myArr2 = new int[5];          //line 2
int myArr3[]; myArr3 = new int[5];   //line 3
int myArr4[]; myArr4[] = new int[5]; //line 4
```

(1) line 1

(2) line 2

(3) line 3

(4) line 4

(5) All the lines of code are correct.

Question 31 **(1)**

Study the code below and indicate what the value of `val` will be:

```
int val = 5;
int result = val++ * 10;
```

(1) 0

(2) 15

(3) 50

(4) 60

(5) None of the above

Study the code in SECTION 17 and answer questions 32 and 33.

CODE SECTION 17

```
String canAdvance = "No";
String gameCode = "Tetris";
int levelComplete = 1;

//--i--
boolean nextLevel = levelComplete >= 1;
boolean play = (game.substring(3,4).equals("i") && 
nextLevel);

if (play) canAdvance = "Yes";
//----
```

Question 32 **(1)**

Indicate what the values of `nextLevel`, `play` and `canAdvance` will be.

(1) `Error`

(2) `nextLevel: true`
`play: false`
`canAdvance: true`

(3) `nextLevel: true`
`play: true`
`canAdvance: Yes`

(4) `nextLevel: true`

`play: true`

`canAdvance: Yes`

(5) `nextLevel: false`

`play: false`

`canAdvance: No`

Question 33 **(1)**

True or false? The next section of code can replace the code in section (i) of CODE SECTION 17 and still give the same result?

```
boolean nextLevel = false;
boolean play = false;
if (levelComplete >=1) {nextLevel = true;}
if (gameCode.substring(3,4).equals("i")) {play = true;}
if (nextLevel && play) {canAdvance = "Yes";}
```

(1) True

(2) False

Question 34 **(1)**

Study the code below and indicate what the values will be of `totalPoints`, `place` and `hasMore`.

```
boolean hasMore = true;
int[] points = {1,2,4,3,2,1};
int place = points.length - 1, totalPoints = 0;
while (hasMore)
 {
  totalPoints = totalPoints + points[place];
  place--;
  if (place < 0) hasMore = false;
 }
```

(1)
```
totalPoints: 12
place: 0
hasMore: false
```

(2)
```
totalPoints: 13
place: -1
hasMore: false
```

(3)
```
totalPoints:1
place: 4
hasMore: false
```

(4)
```
totalPoints: 13
place: 0
hasMore: false
```

(5)
```
totalPoints: 13
place: -1
hasMore: true
```

Study the code in CODE SECTION 18 and answer questions 35 to 38 that follow:

CODE SECTION 18

```
String[] services = {"to let","to buy"};
String[] types =
```

```
{"house","flat","townhouse","commercial","land",
         "farm","development"};
char[] spaces = {'1','2','3','4','5','6'};
char[] garages = spaces;
char[] parking = spaces;
String bathrooms = new String(spaces);

String[] place = new String[types.length];
//--(i)--

String mess = null;
for (int i = 0; i < garages.length; i++){
  mess = mess + garages[i];
}
```

Question 35 **(1)**

Indicate which one of the following options can replace the array declaration that allocates the values to the array `spaces`.

(1)
```
char[] spaces = new char[6];
for (int i=0; i<6; i++){
  spaces[i] =(char)(i+1);
}
```

(2)
```
char[] spaces = new char[6];
for (int i=0; i<6; i++){
  spaces[i] = char.valueOf(i+1);
}
```

(3)
```
char[] spaces = new char[6];
for (int i=0; i<6; i++){
  String val = Integer.toString(i+1);
  spaces[i] = val.chars();
}
```

(4)
```
char[] spaces = new char[6];
for (int i=0; i<6; i++){
  String val = Integer.toString(i+1);
  spaces[i] = val.charAt(0);
}
```

(5) None of the above

Question 36 **(1)**

Indicate which one of the following options can replace the array declaration that allocates the values to the array `services`.

(1)
```
String[] services = new String[2];
services[0] = "to let";
services[1] = "to buy";
```

(2)
```
String[] services = new String[2];
services[1] = "to let";
services[2] = "to buy";
```

(3)
```
String[] services = new String[1];
services[0] = "to let";
services[1] = "to buy";
```

(4)
```
String services[2] = new String[];
services[0] = "to let";
services[1] = "to buy";
```

(5)
```
String services[2] = new String[];
services[1] = "to let";
services[2] = "to buy";
```

Question 37 **(1)**

Indicate which one of the following options will initialize the array `place` by correctly combining the contents of arrays `types` and `services` .

The final output for the array `place` should be:

```
house:to let:to buy
```

```
flat:to let:to buy

townhouse:to let:to buy
```

etc....

(1)
```
for (int i=0; i < place.length; i++){
 place[i] = types[i] + ":" + services[0];
 place[i] = ++services[1];
}
```

(2)
```
for (int i=0; i < place.length; i++){
 place[i] = types[i] + ":" + services[0] + ":" +
services[1];
}
```

(3)
```
for (int i=0; i < place.length; i++){
 place[i] = types[i] + ":" + services[0];
 place[i] = types[i] + ":" + services[1];
}
```

(4)
```
for (int i=0; i < place.length; i++){
 place[i] = types[i] + ":" + services[0];
 place[i] = services[1];
}
```

(5) None of the above

Question 38 **(1)**

Indicate what the value of `mess` will be.

(1) Error message. You cannot add a `char` value to a `String` value.

(2) `123456`

(3) `null123456`

(4) `['1', '2', '3', '4', '5, '6'']`

(5) [123456]

Question 39 **(1)**

Which one of the following is **NOT** an example of an *exception* error that can occur in Java?

(1) A user entered invalid data.

(2) A file that needs to be opened cannot be found.

(3) A network connection has been lost in the middle of communications.

(4) The JVM has run out of memory.

(5) The programmer entered the incorrect code to create a button.

Study the code in CODE SECTION 19 and answer questions 40 to 43 that follow.

CODE SECTION 19

```
public class Utils {


public double subtotal(double a, double b) {
            return a+b;
}


public int subtotal(int a, int b){
          return a+b;
}


public int tip(double answerOne) {
          return (int) (answerOne * .10);
}


public float total(double answerOne, int answerTwo) {
        return (float) (answerOne + answerTwo);
}


public double total(double answerOne, double answerTwo)
{
        return (float) (answerOne + answerTwo);
```

```
}

public String display(String mess){
          return mess.toUpperCase();
}

public String toString(String mess, double amount){
          return mess + " " + amount;
}

}//Utils


Utils u = new Utils();

int val1 = 40; int val2 = 80;

double answerOne = u.subtotal(val1,val2);
int answerTwo = u.tip(answerOne);
int answerThree = (int) u.total(answerOne,answerTwo);
String mess = u.toString("tip: R", answerTwo);
u.display(mess);
```

Question 40 **(1)**

Indicate what the value of `answerOne` will be.

(1) 0

(2) 120

(3) 120.0

(4) Error message – there are more than one method `subtotal()` in the class `Utils`.

(5) None of the above.

Question 41 **(1)**

Indicate what the value of `answerTwo` will be.

(1) 0

(2) 12

(3) 12.0

(4) Error message. You cannot cast a double value to an integer.

(5) None of the above.

Question 42 **(1)**

Indicate what the value of `answerThree` will be.

(1) 0

(2) 132

(3) 132.0

(4) Error message. You cannot cast a float to an integer.

(5) Error message – there is more than one method `total()` in the class `Utils`.

Question 43 **(1)**

Indicate what the value of `mess` will be

(1) tip: R 12

(2) tip: R 12.0

(3) TIP: R 12

(4) TIP: R 12.0

(5) Error message. There is not a method that matches:

`mess(String, int)` in the class `Utils`.

Study the business rules and code in CODE SECTION 20 and answer question 44 that follows.

In the agency the rules are enforced when a person applies to rent a hotel room oversees:

- The person must have a valid Passport

- A female applicant must be 23 years and older
- A male applicant must be 25 years and older
- The person may not have any outstanding criminal record

CODE SECTION 20

```
boolean passport = getPassport();

int age = getAge();

String MF = getMF();

boolean criminal = getCrim();

boolean valid = false;
```

Question 44 **(1)**

Which one of the following expression will **NOT** enforce the rules above?
Assume that all the variables used in the options are declared and initialized correctly.

(1)
```
valid =

 ((passport ==true) && (criminal==false) &&

  (((age>=23)  &&  (MF=="F"  ))|||(  (age>=25)  &&  (MF  ==
"M"))));
```

(2)
```
valid = ((passport ==true) && (criminal==false) &&

          (((age>=23) && (MF.equals("F") ))||

           ((age>=25) && (MF.equals("M")))));
```

(3)
```
valid = ((passport) && !(criminal) &&

          (((age>=23) && (MF.equals("F") ))||

           ((age>=25) && (MF.equals("M")))));
```

(4)
```
valid = ((passport ==true) && (criminal==false) &&

          (((age>=23) && (MF.equals("F") ))||

           ((age>=25) && (MF.equals("M")))));
```

(5)
```
valid = ((passport =true) && (criminal=false) &&

           (((age>=23) && (MF="F"))||

           ((age>=25) && (MF = "M"))));
```

Question 45 **(1)**

The user enters the code below, but an error message is returned. The error is an example of a ____ type of error.

```
int[] myArray = {1,2,3};

myArray[4] = 4;
```

(1) syntax

(2) logic

(3) unchecked exception

(4) checked exception

(5) null-point exception

Question 46 **(1)**

Which one of the following statements describes **inheritance** correctly?

(1) It allows generic code to be placed in a superclass and more specialized code in subclasses, thus promoting code reuse.

(2) It is used when a subclass is more general than a superclass thereby creating a class hierarchy.

(3) The primary reason for using it is to reduce execution times of programs.

4) Using it reduces errors because you can simply copy and paste code from a superclass to new subclasses.

(5) The primary reason for using it is to make the code more “human” readable.

Question 47 **(1)**

Which one of the following correctly illustrates what a superclass is?

(1) …a more general class from which other classes derive their methods and data.

(2) …a more specific class that derives or inherits from another class.

(3) …a set of classes defined as a set of fields.

(4) …a child that looks and act like their parents.

(5) None of the above.

Question 48 **(1)**

___ is the capability of an object to have data and functionality available to the user, without the user having to understand the implementation within the object.

(1) Aggregation

(2) Encapsulation

(3) Inheritance

(4) Polymorphism

(5) Visibility

Question 49 **(1)**

Which one of the following statements explains the contents of `activity_main.xml` file in an Android application project?

(1) It contains information about the sound and image files used in the application.

(2) It contains information about the default Graphical User Interface of the application.

(3) It contains the code of the class `MainActivity`.

(4) It contains a summary of all the setup information of the application.

(5) It contains a summary of all the XML files used in the application.

Question 50 **(1)**

In an Android project the _____ folder includes the resource files (images, music and videos) for the project.

(1) `assets`

(2) `bin`

(3) `gen`

(4) `res`

(5) `src`

TOTAL: 50