COMP 2522 : Midterm Exam (OOP using Java)

February 29th 2024

1. The exam is closed book. You may not use notes, texts, manuals during exam.
2. Cheating will be results in disqualification from the exam and possibly expulsion from the course.
3. No talking during exam. Please raise your hand if you have a question and wait for an instructor.
4. There are 16 pages. Make sure you have all the pages **BEFORE** you start the exam.
5. There are 4 parts to the exam. Section A is a multiple choice (8 marks); Section B is Give the output using multiple choices (16 marks); Section C is Find the error (4 marks); Section D is coding (6 marks). Max 34 marks. Make sure your FULL NAME and **SET** is on the exam.
6. LISTEN TO THE INSTRUCTOR FOR FURTHER INSTRUCTIONS AND POSSIBLE CHANGES DURING THE EXAM. ALSO CHECK THE PROJECTOE DISPLAY OR WHITEBOARD.
7. Time limit : **60 minutes**.
8. Good luck !

1 Part A : Multiple choice (8 marks). Circle the best answer. 1 mark each

1. Interfaces allows for :
   (a) Multiple type matching
   (b) Multiple method declaration without conflict
   (c) Constants to be defined
   (d) all of the above
   (e) none of the above

```
1 public class Head{
2         Brain brain;
3         private class Brain{}
4
5         public static void main(String[] args){
6                 Head h = new Head();
7         }
8 }
```

2. The code to instantiate a Brain in method main() is :
   (a) Brain brain = new Brain();
   (b) h.brain = new Brain();
   (c) h.brain = h.new Brain();
   (d) Head.Brain brain = new Brain();
   (e) cannot be made, there is no constructor for a Brain object

```
1 package midterm;
2 public class A{
3         private void snafu(){}
4         void foo(){}
5         protected void bar(){}
6 }
```

```
1 package finalexam;
2 import midterm.A;
3 public class B extends A{}
```

```
1 package midterm;
2 public class C{
3         A a = new A();
4 }
```

3. Using the above declarations, what methods can class B access from its parents :

   (a) foo();
   (b) bar();
   (c) snafu();
   (d) foo(); and bar();
   (e) foo(); bar(); and snafu();

4. Using the above declarations, what methods can be accessed usign reference "**a**" :

   (a) a.foo();
   (b) a.bar();
   (c) a.snafu();
   (d) a.foo(); and a.bar();
   (e) a.foo(); , a.bar(); and a.snafu();

5. Using the above declaration, in class B what **modifications** are allowed to the access modifiers to EXPLICITLY OVERRIDE methods from A? **Assume class B definition was moved to the same package "midterm" as class A.**

   (a) void foo() changed to protected void foo(), protected void bar() changed to public void bar()
   (b) void foo() changed to private void foo(), protected void bar() changed to void bar()
   (c) private snafu() changed to public void snafu()
   (d) all of the above are valid
   (e) none of the above are valid

```
1 String[] names = {"hello","goodbye"};
2 Object[] ptr = names;
3 ptr[1] = 12.5;
```

6. The above is :
   (a) compile time error line 2
   (b) runtime error line 2
   (c) compile time error line 3
   (d) runtime error line 3
   (e) allowed because the two arrays are related through inheritance

```
1  interface Transaction{
2          public void run(){}
3  }
4
5  class BankTransaction implements Transaction{
6          public void run(){}
7  }
8
9  class Test{
10         public static void task(ArrayList<Transaction> list){
11                 for(Transaction trans : list)
12                         trans.run();
13         }
14
15         public static void main(String[] args){
16                 ArrayList<BankTransaction> t = new ArrayList<>();
17                 task(t);
18         }
19 }
```

7. The above code :
   (a) will NOT compile, safest fix is task(ArrayList <X>list)
   (b) will NOT compile, safest fix is task(ArrayList <? extends Transaction>list)
   (c) will NOT compile, safest fix is task(ArrayList list)
   (d) will NOT compile, safest fix is task(ArrayList <X implements Transaction>list)
   (e) will compile and run as expected

8. Benefit(s) of using Generics are :
   (a) casting not required on returned values
   (b) strong type checking during compile time rather than runtime
   (c) methods can be used with any type
   (d) all of the above
   (e) none of the above

2 Part B : Give the output. All the code here compiles and runs (16 marks)

```
1  class Tansaction{
2          public void amount(){
3                  System.out.println("amount");
4          }
5
6          public Transaction(){
7                  System.out.println("Transaction created");
8                  amount();
9          }
10 }
11
12 class Debit extends Transaction{
13         public void amount(){
14                 System.out.println("debit amount");
15         }
16
17         public Debit(){
18                 System.out.println("Debit created")
19                 amount();
20         }
21 }
22
23 public class Withdrawal extends Debit{
24         int fee = 2;
25         public void amount(){
26                 System.out.println("withdrawal and fee "+fee);
27         }
28
29         public Withdrawal(int n){
30                 System.out.println("Withdrawal created with a fee of " + fee);
31                 fee = n;
32                 System.out.println("fee="+fee);
33         }
34
35         public static void main(String[] args){
36                 Debit d = new Withdrawal(4);
37         }
38 }
```

9. Give the **output** for the above code : (2 mark)

   (a) Debit created, debit amount, Transaction created with a fee of 2, fee=4

   (b) Transaction created, amount, Debit created, debit amount, Transaction created with a fee of 2, fee=4

   (c) Transaction created, withdrawal and fee 2, Debit created, withdrawal and fee 2, Withdrawal created with a fee of 2, fee=4

   (d) Transaction created, amount, debit amount, withdrawal and fee of 2, Debit created, debit amount, withdrawal and fee 2, Withdrawal created with fee of 2, fee=4

   (e) Transaction created, withdrawal and fee 0, Debit created, withdrawal and fee 0, Withdrawal created with a fee of 2, fee=4

```
1  Class Cycle{
2          int numWheels = 1;
3          int wheelsize;
4          Cycle(){
5                  output("number of wheels = " + numWheels + ", wheel size = "+
                       wheelsize);
6                  wheelsize = 24;
7          }
8
9          static int age = output("Cycle.age in years initialized");
10
11         static int output(String s){
12                 System.out.println(s);
13                 return 2;
14         }
15 }
16
17 class MountainBike extends Cycle{
18         int numWheels = Cycle.output("MountainBike.numWheels initialized");
19         MountainBike(){
20                 Cycle.output("snumWheels = " + numWheels);
21                 Cycle.output("wheel size = " + wheelsize);
22         }
23
24         static int hydrolicBrakes = Cycle.output("MountainBike.hydrolicBrakes
               initialized");
25 }
26
27 public class Trail{
28         public static void main(String[] args){
29                 System.out.println("Started program");
30                 MountainBike m = new MountainBike();
31         }
32 }
```

10. Give the **exact output** for the above class : (2 marks)

    (a) Started program, MountainBike.hydrolicBrakes initialized, MountainBike.numWheels initialized, number of whells = 1, wheel size = 0, Cycle.age in years initialized, snumWheels = 1, wheel size = 24

    (b) Started program, Cycle.age in years initialized, MountainBike.hydrolicBrakes initialized, number of wheels = 1, wheel size = 0, snumWheels = 1, wheel size = 24

    (c) Cycle.age in years initialized, MountainBike.hydrolicBrakes initialized, Started program, number of wheels = 0, wheel size = 0, MountainBike.numWheels initialized, snumWheels = 1, wheel size = 24

    (d) Cycle.age in years initialized, MountainBike.hydrolicBrakes initialized, Started program, number of wheels = 1, wheel size = 0, snumWheels = 1, wheel size = 24

    (e) Started program, Cycle.age in years initialized, MountainBike.hydrolicBrakes initialized, number of wheels = 1, wheel size = 0, MountainBike.numWheels initialized, snumWheels = 2, wheel size = 24

```
1  public class Swapper{
2          public static <T> void swap(T a, T b){
3                  T temp = a;
4                  a = b;
5                  b = temp;
6          }
7
8          public static <T> T swap(T a, T b, T c){
9                  T temp = a;
10                 a = b;
11                 b = c;
12                 c = temp;
13                 return c;
14         }
15
16         public static <T> void swap(T[] pool, int x, int y){
17                 T temp = pool[x];
18                 pool[x] = pool[y];
19                 pool[y] = temp;
20         }
21
22         public static void main(String[] args){
23                 String a = "hello";
24                 String b = "goodbye";
25                 String c = "fubar";
26                 Integer[] collection = {1,2,3,4,5};
27                 swap(a,b);
28                 System.out.println(a);
29                 a = "hello";
30                 b = "goodbye";
31                 c = "fubar";
32                 c = (String)swap(a,b,c);
33                 System.out.println(c);
34                 swap(collection,2,3);
35                 System.out.println(collection[3]);
36
37         }
38 }
```

11. Give the **exact output** for the above code : (2 marks)

    (a) hello, hello, 3

    (b) hello, fubar, 4

    (c) goodbye, hello, 2

    (d) hello, hello, 2

    (e) hello, goodbye, 2

```
interface Car{
        void start();
}

class Ford implements Car{
        public void start(){
                System.out.println("F");
        }
}

class GM extends Ford implements Car{
        public void start(){
                System.out.println("G");
        }
}

class Chrystler extends Ford implements Car{
        public void start(){
                System.out.println("C");
        }
}

class Ram extends GM{
        public void start(){
                super.start();
                System.out.println("R");
        }
}

public class ParkingLot{
        public static void main(String[] args){
                Car x;
                x = new Ford();
                x.start();
                x = new GM();
                x.start();
                x = new Chrystler();
                x.start();
                x = new Ram();
                x.start()
        }
}
```

12. Give the output for the above code : (2 marks)

    (a) F, G, C, R

    (b) F, F, G, F, C, F, R

    (c) F, F, F, G, F, C, G, F

    (d) F, G, C, G, R

    (e) syntax error, code line "Car x;" is illegal

```
1  public class Experiment{
2          public static void heat() throws Exception{
3                  if(Math.random(2)==1)
4                          throw new Exception();
5          }
6
7          public static void test() throws Exception{
8                  try{
9                          heat();
10                         System.out.println("matter heated");
11                 } catch (Exception e) {
12                         System.out.println("exception occurred");
13                 } finally {
14                         System.out.println("clean up site");
15                 }
16                 System.out.println("end of test");
17         }
18
19         public staic void main(String[] args){
20                 Experiment e = new Experiment();
21                 e.test();
22         }
23 }
```

13. Give the output of the above code **IF method heat() <u>throws</u> an exception**: (2 marks)

    (a) matter heated, exception occurded, clean up site

    (b) exception occurred, matter heated, clean up site, end of test

    (c) exception occurred, clean up site

    (d) exception occurred, end of test

    (e) exception occurred, clean up site, end of test

14. Give the output of the above code **IF method heat() <u>DOES NOT</u> throw an exception**: (2 marks)

    (a) matter heated, clean up site

    (b) matter heated, clean up site, end of test

    (c) matter heated, end of test

    (d) end of test

    (e) matter heated, end of test, clean up site

```
1  class WarpException extends Exception{}
2  class DissolveExcepion extends Exception{}
3
4  class Morpher{
5          public static void warp() throws WarpException{
6                  throw new WarpException();
7          }
8
9          public static void dissolve() throws DissolveException{
10                 throw new DissolveException();
11         }
12
13         public static void main(String[] args){
14                 try{
15                         warp();
16                         System.out.println("warped");
17                 } finally {
18                         System.out.println("dissolve applied");
19                         dissolve();
20                 }
21                 System.out.println("finished");
22         }
23 }
```

15. Give the output for the above code : (2 marks)

    (a) WarpException, warped, dissolve applied, DissolveException, finished

    (b) dissolve applied, WarpException, DissolveException

    (c) dissolve applied, DissolveException

    (d) dissolve applied, finished

    (e) warped, dissolve applied, finished, WarpException, DissolveException

```
class Cup{
        Cup(int marker){
                System.out.println("Cup("+marker+")");
        }

        void f(int marker){
                System.out.println("f("+marker+")");
        }
}

class Cups{
        static Cup c1 = new Cup(1);
        static Cup c2 = new Cup(2);

        Cups(){
                System.out.println("Cups()");
        }
}

public class ExplicitStatic{
        public static void main(String[] args){
                System.out.println("Inside main()");
                new Cups();
                Cups.c1.f(99);
        }
        static Cups x = new Cups();
        static Cups y = new Cups();
}
```

16. Give the output of the above code : (2 marks)

    (a) Inside main(), Cups(), f(99)

    (b) Cups(), Cups(), Cup(1), Cup(2), Inside main(), Cups(), f(99)

    (c) Inside main(), Cups(1), Cup(2), Cups(), Cups, Cups(), f(99)

    (d) Cups(), Cup(1), Cup(2), Cups(), Inside main(), Cups(), f(99)

    (e) Cup(1), Cup(2), Cups(), Cups(), Inside main(), Cups(), f(99)

3 Part C : Error code. Code here has one error. Explain why the code is in error and provide a fix so that the intent of the author remains. (4 marks)

```
1  class Container{        // DO NOT CHANGE THIS CLASS
2          protected String id;
3          public Container(String n){
4                  id = n;
5          }
6  }
7
8  public class ShippingContainer extends Container{
9          int size;
10         public ShippingContainer(String n){     // COMPILE ERROR HERE
11                 id = n;
12         }
13         public void setSize(int s){
14                 size = s;
15         }
16         public static void main(String[] args){ // DO NOT CHANGE MAIN
17                 ShippingContainer box = new ShippingContainer("456-123");
18         }
19 }
```

17. The above code doesn't **compile**. Explain **why and fix the code** so it compiles and runs **as expected**. (2 marks)

```
1  public class Table{
2          public static void addNumbers(List<Integer> list){
3                  for(int i=1; i<=10 ; i++)
4                          list.add(i);
5          }
6
7          public static void main(String[] args){
8                  List<Integer> test = newArrayList<>();
9                  addNumbers(test);
10                 List<Number> data = new ArrayList<>();
11                 addNumbers(data);        // ERROR
12         }
13 }
```

18. The above code doesn't compile. Explain why and fix the code so it will compile and run as expected. (2 marks)

4 Part D : Coding (6 marks)

```
1  class Agent{
2          public void spy(){
3                  System.out.println("spy");
4          }
5  }
6  class Gambler{
7          public void gamble(){
8                  System.out.println("gamble");
9          }
10 }
11 class Killer{
12         public void kill(){
13                 System.out.println("bang");
14         }
15 }
16 class JamesBond{}
17
18 // Do not change any of this code
19 public class Spectre{
20         public void sneak(Agent a){
21                 a.spy();
22                 System.out.println("found secret");
23         }
24         public void cardGame(Gambler g){
25                 g.gamble();
26                 System.out.println("broke the bank");
27         }
28         public void getSpectre(Killer k){
29                 k.kill();
30                 System.out.println("dead");
31         }
32
33         public static void main(String[] args){
34                 Spectre g = new Spectre();
35                 JamesBond d = new JamesBond();
36                 g.sneak(b);
37                 g.cardGame(b);
38                 g.getSpectre(b);
39                 System.out.println("Crime does not pay");
40         }
41 }
```

19. Change the code above so that JamesBond can be passed to all the methods in class Spectre so crime will not win! **Note: Agent, Gambler, Killer are NOT similar types (MUTUALLY EXCLUSIVE!)**. Hint : a JamesBond is-a Agent, JamesBond is-a Gambler, JamesBond is-a Killer BUT **"Agent is-a Gambler" is NOT TRUE etc.** (2 marks)

Answer :

```
1  interface Iterator{
2          boolean hasNext();
3          int next();
4  }
5
6  public class List{
7          int pos = 0;
8          int[] data;
9
10         class MyIterator implements Iterator{
11                 public int next(){
12                         return data[pos++];
13                 }
14                 public boolean hasNext(){
15                         return ((pos>0) && (pos<data.length));
16                 }
17         }
18 }
```

20. Give the generic version of the code above. (3 marks)