Πανεπιστήμιο Κρήτης, Τμήμα Επιστήμης Υπολογιστών
HY252 – Αντικειμενοστεφής Προγραμματισμός
Εξάμηνο: **Χειμερινό 2014-2015**
Διδάσκων: **Γιάννης Τζίτζικας**

Θέμα	Βαθμός
1 (10μ)	
2 (20μ)	
3 (10μ)	
4 (20μ)	
5 (15μ)	
6 (25μ)	
Σύνολο (100)	

Εξέταση Προόδου

(ΕΝΔΕΙΚΤΙΚΕΣ ΛΥΣΕΙΣ κ' ΣΥΧΝΑ ΛΑΘΗ)

15 Νοεμβρίου 2014, 10πμ-12πμ (2 ώρες)

Όνομα	
Επώνυμο	
ΑΜ	
Έτος	

Θέμα 1 (10 μονάδες) // **υλοποίηση απλού αλγορίθμου, πίνακες**

Γράψτε ένα πρόγραμμα, ας το πούμε Simple, το οποίο θα λαμβάνει από την γραμμή εντολών έναν ακέραιο Κ. Εν συνεχεία θα ζητάει από το χρήστη να του δώσει Κ ακεραίους. Τους αριθμούς αυτούς πρέπει να τους αποθηκεύσει σε έναν πίνακα ακεραίων μεγέθους Κ. Στη συνέχεια το πρόγραμμα θα διατρέχει τον πίνακα και θα υπολογίζει και στο τέλος θα εκτυπώνει στην κονσόλα τα εξής:
α) τον ελάχιστο αριθμό
β) το μέγιστο αριθμό
γ) το μέσο όρο των αριθμών
Εν συνεχεία θα αντικαθιστά τον αριθμό του κάθε κελιού του πίνακα με το τετράγωνό του, και μετά θα ξαναδιατρέχει τον πίνακα, θα υπολογίζει και θα εκτυπώνει:
α) τον ελάχιστο αριθμό
β) τον μέγιστο αριθμό
γ) τον μέσο όρο των αριθμών

Ένα παράδειγμα λειτουργίας αν δώσουμε την εντολή java Simple 3 ακολουθεί:

```
Δώσε μου έναν ακέραιο
8
Δώσε μου έναν ακέραιο
9
Δώσε μου έναν ακέραιο
10
Ελάχιστος : 8
Μέγιστος  : 10
Μέσος Όρος: 9.0
Ελάχιστος : 64
Μέγιστος  : 100
Μέσος Όρος: 81
```

Μπορείτε να διαβάσετε την είσοδο και να πάρετε έναν αριθμό με:

Scanner in = new Scanner(System.in);
int aNumber = in.nextInt();

ΕΝΔΕΙΚΤΙΚΗ ΛΥΣΗ

```
package year2015.proodos;

import java.util.Scanner;

class SimpleOperations {
	public static void main(String[] kstr) {
	   int k = Integer.parseInt(kstr[0]);
	  //int k =3;
		Scanner in = new Scanner(System.in);
		int pinakas[] = new int[k];

		for (int i=0;i<k;i++) {
		    System.out.println("Δώσε μου έναν ακέραιο ");
		      int aNumber = in.nextInt();
		      pinakas[i]=aNumber;
		}

		int min = pinakas[0];
		int max = pinakas[0];
		int sum = 0;
		for (int i=0;i<k;i++) {
			if (pinakas[i] < min) min = pinakas[i];
			if (pinakas[i] > max) max = pinakas[i];
			sum+=pinakas[i];
		}
		System.out.println("Ελάχιστος : " + min);
		System.out.println("Μέγιστος  : " + max);
		System.out.println("Μέσος Όρος: " + (((float)sum)/k));


		//---
		for (int i=0;i<k;i++)
			pinakas[i]=pinakas[i]*pinakas[i];
		//--


		min = pinakas[0];
		max = pinakas[0];
		sum = 0;
		for (int i=0;i<k;i++) {
			if (pinakas[i] < min) min = pinakas[i];
			if (pinakas[i] > max) max = pinakas[i];
			sum+=pinakas[i];
		}
		System.out.println("Ελάχιστος : " + min);
		System.out.println("Μέγιστος  : " + max);
		System.out.println("Μέσος Όρος: " + sum/k);
	}

}
```

```
Βαθμολόγηση
3 Το μέγεθος του πίνακα να ορίζεται από το command line arg
2 Γέμισμα πίνακα
2 Εύρεση min/max/avg
3 Τετραγωνισμός
```

Θέμα 2 (20 μονάδες)
α) [15 μονάδες]
Γράψτε τι θα εκτυπωθεί στην κονσόλα αν δώσουμε στη γραμμή εντολών `java Program`. Προσέξτε τη σειρά.

```
class A {
	int p1=1;
	int p2=1;
	A(int p1, int p2) {
		this.p1 = p1;
		this.p2 = p2;
	}
	A() {
		this(2,2);
	}

}

class B extends A {

	B(){
		super(3,3);
	}
	B(int p1, int p2) {

	}
	B(int p1, int p2, int k) {
		for (int i=0; i<k; i++){
			System.out.println(p1 + " " + p2);
		}
		this.p1=9;
		this.p2=9;
	}
	public String toString() {
		return p1 + " " + p2;
	}
}

class Program {
	public static void main(String lala[]) {
		B b1 = new B();
		B b2 = new B(4,4);
		B b3 = new B(10,10,3);
	    System.out.println(b1);
	    System.out.println(b2);
	    System.out.println(b3);
	}
}
```

ΕΝΔΕΙΚΤΙΚΗ ΛΥΣΗ &Βαθμολόγηση
Α) [15 μονάδες]

10 10
10 10
10 10 // 3 μονάδες για τις τρεις παραπάνω σειρές με τα ζεύγη δεκαριών
3 3   // 4 μονάδες
2 2   // 4 μονάδες
9 9   // 4 μονάδες

ΣΥΧΝΑ ΛΑΘΗ
1. Στο πρώτο ερώτημα. Το 20% περίπου το έκαναν σωστό που δείχνει πολλά προβλήματα σχετικά με την κληρονομικότητα.
2. Στο δεύτερος ερώτημα οι περισσότεροι έδωσαν σωστή απάντηση σχετικά με την μεταβλητή static αλλα έκαναν λάθος στον constructor.

β) [5 μονάδες]

Θέλουμε να εμπλουτίσουμε τον παραπάνω κώδικα ώστε η κλάση Α να γνωρίζει πόσα στιγμιότυπα της έχουν δημιουργηθεί. Δώστε τον κώδικα που πρέπει να προστεθεί και πού, και δώστε και την εντολή που πρέπει να προστεθεί στη main της Program ώστε να εκτυπώνεται το πλήθος των στιγμιότυπων της Α. Επίσης πείτε τι θα εκτυπώσει αυτή η εντολή.

ΕΝΔΕΙΚΤΙΚΗ ΛΥΣΗ
Προσθέτουμε ένα static instance variable στην Α:

```
static int numOfInstances =0;
```

Το αυξάνουμε στον constructor (int,int) που απ'ότι φαίνεται καλείται και από τον άλλο constructor (αν εκτελείτε σε κάθε δημιουργία αντικειμένου), άρα η νέα του μορφή είναι:

```
A(int p1, int p2) {
            this.p1 = p1;
            this.p2 = p2;
            numOfinstances++; // gia skelos (B)
      }
```

Προσθέτουμε στην main της Program το System.out.println(A.NumOfInstances).

Θα εκτυπώσει 3.

Βαθμολόγηση
2 μονάδες για το static instance variable
2 μονάδες για σωστό εμπλουτισμό του σωστού constructor
1 μονάδα για το System.out.println(A.NumOfInstances) και την τιμή του

Θέμα 3 (10 μονάδες) // **abstract classes, subtyping**

Ο παρακάτω κώδικας έχει 3 λάθη, δηλαδή προβλήματα που αποτρέπουν την μεταγλώττιση (compile) του κώδικα. Εντοπίσετε τα και εξηγήστε γιατί είναι λάθος

```
abstract class E1 {
   protected int pedio = 20;
   abstract  int superFunction1();
   abstract  int superFunction2();
   abstract  int superFunction3();
}

class E2 extends E1 {
      private int pedio = 30;
      int superFunction1() {return pedio;}
      int superFunction2() {return ((E1)this).pedio;}
}

class E3 extends E2 {
      private int pedio = 40;
      public static void main(String a[]) {
            E1 e1 = new E1();
            E2 e2 = new E2();
            E3 e3 = new E3();
            e1 = e3;
            e3 = e2;

      }
}
```

ΕΝΔΕΙΚΤΙΚΗ ΛΥΣΗ

1. Η E2 αφού είναι συγκεκριμένη θα έπρεπε να δίνει υλοποίηση σε όλες τις abstract που κληρονομεί, άρα και στην superFunction3();
2. Δεν μπορούμε να δημιουργήσουμε στιγμιότυπο μιας abstract κλάσης, άρα το = new E1() είναι λάθος
3. Δεν μπορούμε σε μια μεταβλητή τύπου X να εκχωρήσουμε ένα αντικείμενο που δεν είναι X (ή υποκλάση του X), άρα η εντολή e3=e2 είναι λάθος

Βαθμολόγηση

3 μονάδες για κάθε ένα πρόβλημα που εντοπίστηκε(+1 μονάδα σε όποιον τα βρήκε και τα 3).

Αν κάποιος σημείωσε ως λάθος κάτι που είναι σωστό, να έχει μείωση βαθμού.

ΣΥΧΝΑ ΛΑΘΗ

1. Πολλοί φοιτητές σημείωναν ως λάθος την εντολή e1=e3 ενώ στην ιεραρχία η κλάση E1 είναι πιο ψηλά από την κλάση E3.
2. Άλλο συχνό λάθος ήταν στην εντολή return ((E1)this).pedio , για την οποία κάποιοι φοιτητές πίστευαν πως είναι λάθος

συντακτικά

3. Επίσης υπήρχαν κάποια γραπτά στα οποία αναφερόταν ως λάθος η εντολή E3 e3=new E3() επειδή θα προκαλούσε ατέρμονη επανάληψη, μιας και βρίσκεται μέσα σε αυτή την κλάση η main. Άρα πίστευαν πως δεν πρέπει να φτιάχνονται στιγμιότυπα για την κλάση που περιέχει την main.

4. Ακόμα μερικοί φοιτητές θεωρούσαν πως το λάθος είναι ότι δεν υπήρχαν constructors , ενώ κάποιοι άλλοι πως η E3 έπρεπε να υλοποιεί τις συναρτήσεις της E2.

5. Τέλος , ένα λιγότερο συχνό λάθος ήταν πως δε μπορεί να αλλάξει ο τύπος του int pedio από protected στην κλάση E1 σε private στην κλάση E2.

Θέμα 4 (20 μονάδες) // **overriding, hiding**

Ποια είναι η έξοδος του παρακάτω κώδικα (της main της Test) και γιατί;
Δικαιολογείστε σύντομα όπου κρίνετε ότι χρειάζεται.

```
class European {
  String name ="Anonymous";
  String countryName = "A European Country ";
  String saySomething() { return "Hi";}
  String whereAreYouFrom() { return countryName;}
}

class Greek extends European {
      String countryName = "Greece";
      String saySomething() { return "Γεια";}
      String whereAreYouFrom() { return countryName + "( " +
                                    super.whereAreYouFrom() + ")" ;}
      Greek(String name) {this.name=name;}
}

class Test {
      public static void main(String[] lala){
            Greek g1 = new Greek("Αριστοτέλης");
            System.out.println(g1.whereAreYouFrom());
            System.out.println(g1.saySomething());

            European e1 = g1;
            System.out.println(e1.whereAreYouFrom());
            System.out.println(e1.saySomething());

            System.out.println(e1.countryName);
            System.out.println(g1.countryName);

            European e2 = new Greek("Επιμενίδης");
            System.out.println(e2.name);
            System.out.println(e2.countryName);
            System.out.println(e2.whereAreYouFrom());
      }
}
```

```
ΕΝΔΕΙΚΤΙΚΗ ΛΥΣΗ & Βαθμολόγηση

1.    Greece( A European Country )  // 2 μ.
2.    Γεια                          // 1 μ.
3.    Greece( A European Country )  // 3 μ.
4.    Γεια                          // 3 μ.
5.    A European Country            // 2 μ.
6.    Greece                        // 2 μ.
7.    Επιμενίδης                    // 2 μ.
8.    A European Country            // 2 μ.
9.    Greece( A European Country )  // 3 μ.
```

Θέμα 5 (15 μονάδες) // **exceptions**

Δείτε τον παρακάτω κώδικα. Σχολιάσετε τις υπογραφές της `method1()` και `method2()` της κλάσης `BB` σχετικά με exceptions. Δηλαδή σχολιάστε αν κάθε μία τους είναι σωστή ή λάθος και για ποιο λόγο;

```
class Exception1 extends Exception { }
class Exception2 extends Exception1 { }
class Exception3 extends RuntimeException {}

class AA {
      int method1() throws Exception1 {return 0;      }
      int method2() throws Exception2 {return 0;      }
}

class BB extends AA {
      int method1() throws Exception1 {
            throw new Exception2();
      }

      int method2() {
            if (2<3) { throw new Exception3(); }
            else return 1;
      }
}
```

ΕΝΔΕΙΚΤΙΚΗ ΛΥΣΗ

Η υπογραφή
int method1() throws Exception2
είναι σωστή γιατί

1. Πρέπει να δηλωθεί η εξαίρεση στην επικεφαλίδα της, αφού πετάει exception στο σώμα της
2. Αν και πετάει Exception2 το ότι η υπογραφή δηλώνει E1 είναι σωστό, αφού έχει οριστεί το E2 ως υποτύπος του E1, άρα ένα στιγμιότυπο του E2 είναι και στιγμιότυπο του E1

Η υπογραφή
int method2()
είναι σωστή γιατί

1. Αν και το σώμα πετάει Exception3, δεν είναι απαραίτητη η δήλωση του στην υπογραφή διότι το Exception3έχει δηλωθεί υποκλάση του RuntimeException, που είναι unchecked.
2. Ναι μεν υποσκελίζει την method2 της AA που δηλώνει Exception2, αλλά μια μέθοδος που επικαλύπτει μία άλλη δεν είναι απαραίτητο. Δείτε τους κανόνες.

Βαθμολόγηση
7 μονάδες για την κάθε μέθοδο, +1 εάν είναι σωστές και οι δύο.
Αν η δικαιολόγηση είναι ελλιπής τότε ποσοστιαία βαθμός.
Αν δεν υπάρχει δικαιολόγηση ή λάθη, τότε δεν προσμετράται βαθμός.

ΣΥΧΝΑ ΛΑΘΗ

Η υπογραφή int method1() throws Exception2 είναι σωστή γιατί

1. πρέπει να δηλωθεί η εξαίρεση στην επικεφαλίδα της, αφού πετάει exception στο σώμα της
2. αν και πετάει Exception2 το ότι η υπογραφή δηλώνει E1 είναι σωστό, αφού έχει οριστεί το E2 ως υποτύπος του E1, άρα ένα στιγμιότυπο του E2 είναι και στιγμιότυπο του E1

Συχνά λάθη:

1. Λάθος, γιατί η method1() θα έπρεπε να έχει οριστεί ως int method1() throws Exception2, αφού πετάει Exception2
2. Σωστό, γιατί και το Exception1 και το Exception2 κάνουν extend την κλάση Exception
3. Λάθος, γιατί πετάει πάντα Exception
4. Λάθος, γιατί έχει δηλωθεί ως int και δεν κάνει return πουθενά (δεν ήταν αυτό το ζητούμενο...)

Η υπογραφή int method2() είναι σωστή γιατί

1. Αν και το σώμα πετάει Exception3, δεν είναι απαραίτητη η δήλωση του στην υπογραφή διότι το Exception3έχει δηλωθεί υποκλάση του RuntimeException, που είναι unchecked.
2. Ναι μεν υποσκελίζει την method2 της AA που δηλώνει Exception2, αλλά μια μέθοδος που επικαλύπτει μία άλλη δεν είναι απαραίτητο. Δείτε τους κανόνες.

Συχνά λάθη:

1. Λάθος, γιατί η method2() θα έπρεπε να έχει οριστεί ως int method2() throws Exception3.
2. Λάθος, γιατί η method2() στην AA έχει δηλωθεί ότι πετάει Exception2, όχι Exception3.
3. Λάθος, γιατί πετάει πάντα Exception

Θέμα 6 (25 μονάδες) // ADT

Καλείστε να σχεδιάσετε και να υλοποιήσετε μια κλάση με όνομα Κουμπαράς. Σε έναν κουμπαρά πρέπει να μπορούμε να προσθέσουμε κέρματα (ένα τη φορά). Επίσης ο κουμπαράς πρέπει να προσφέρει μεθόδους που να επιτρέπουν σε κάποιον να μάθει τη συνολική αξία των κερμάτων που περιέχει, καθώς και το πλήθος των κερμάτων που περιέχει. Επίσης πρέπει να προσφέρει μια μέθοδο doBreak(), η οποία να .. τον σπάει. Αν ένας κουμπαράς σπάσει μετά δεν πρέπει να μπορεί ούτε να δεχτεί κάποιο κέρμα ούτε να επιστρέψει το σύνολο της αξίας των κερμάτων που περιέχει ή το πλήθος τους.

α) [10 μονάδες]
ορίστε τις υπογραφές των μεθόδων σε ένα interface με όνομα **I_Koumparas**. Διαχωρίστε τις μεθόδους με βάση το είδος τους (transformers, observers, accessors). Ορίστε τα pre-conditions, και τα post-conditions των μεθόδων και τις αναλλοίωτες συνθήκες (invariants).

β) [10 μονάδες]
Δώστε μια κλάση **Koumparas** που να υλοποιεί το διεπαφή και να σέβεται το συμβόλαιο που περιγράφει.

γ) [5 μονάδες]
Δώστε τον κώδικα που πρέπει να γράψει ένας πελάτης της κλάσης Koumparas, για να δημιουργήσει έναν κουμπαρά, να προσθέτει μέσα του τα εξής κέρματα: ένα μονόευρω, ένα 20λεπτο, ένα 5λεπτο, ένα 2ευρω, και μετά να εκτυπώσει το πλήθος των κερμάτων που έχει ο κουμπαράς και το σύνολο της αξίας τους (στην προκειμένη 3 ευρώ και 25 λεπτά, 4 κέρματα), μετά να τον σπάει, και μετά να προσπαθεί (ανεπιτυχώς) να προσθέσει σε αυτόν ένα νέο κέρμα.

ΕΝΔΕΙΚΤΙΚΗ ΛΥΣΗ

```
class BrokenKoumparasException extends Exception  {
      BrokenKoumparasException(String s) { super(s);}
}

 enum Kerma {     lepto1, lepto2, lepto5, lepto10, lepto20, lepto50,   euro1,
euro2 }

interface I_Koumparas {
      // OBSERVERS

      /* Boolean isBroken()
       * Pre: None
       * Post: True is doBreak has been called at least once
       */
      boolean isBroken() ;


      // TRANSFORMERS

      /*
       * Pre: Koumparas is not broken
       * Post: Koumparas now knows ...
       */

      /*
       * doBreak()
       * Pre: None
       * Post: is broken, the rest accessors will not function
       *    (will throw exceptions)
       */
      void  doBreak();

      /*
       * add_Kerma(Kerma k) throws BrokenKoumparasException;
       * Pre: Not broken
       * Post: getNumberOfCoins() = getNumberOfCoins()@PRE + 1
       *             getTotalValue() =  getTotalValue()@PRE + k.value
       */
      void addKerma(Kerma k) throws BrokenKoumparasException;

      /* add_Kerma_Euro(int e);
       * Pre: Not broken
       * Post: returns the total number of  coins that have been added
       */


      int getNumberOfCoins() throws BrokenKoumparasException;

      /*
       * Pre: Not broken
       * Post: returns the total value of the coins that have been added
       */

      float getTotalValue() throws BrokenKoumparasException;

}


 class Koumparas implements I_Koumparas {
       private boolean broken=false ;
       private int numberOfcontainedCoins =0;
       private float totalValue =0 ;

       public boolean isBroken() {return broken;}
       public void doBreak() { broken=true;}
```

```
       public void addKerma(Kerma k) throws BrokenKoumparasException {
            if (isBroken())
                  throw new BrokenKoumparasException("Δεν μπορώ να δεχτώ άλλο
κέρμα. Έχω σπάσει!");
            numberOfcontainedCoins++;
            float kermavalue=0;
            switch (k) {
                  case lepto1: kermavalue= 0.01f;
                  break;
            case euro1: kermavalue= 1;
                  break;
            case euro2: kermavalue= 2;
                  break;
            case lepto10: kermavalue= 0.1f;
                  break;
            case lepto2: kermavalue= 0.02f;
                  break;
            case lepto20: kermavalue= 0.2f;
                  break;
            case lepto5: kermavalue= 0.05f;
                  break;
            case lepto50: kermavalue= 0.5f;
                  break;
            }

            totalValue+=kermavalue;
       }



      public int getNumberOfCoins() throws BrokenKoumparasException {
            if (isBroken())
                  throw new BrokenKoumparasException("Δεν μπορώ να σου πως πόσα
κέρματα έχω επειδή έχω σπάσει!");
            return numberOfcontainedCoins;
      }

            /*
             * Pre: Not broken
             * Post:
             */

      public float getTotalValue() throws BrokenKoumparasException {
            if (isBroken())
                  throw new BrokenKoumparasException("Δεν μπορώ να ενημερώσω για
την αξία αφού έχω σπάσει!");
            return totalValue;
      }
}

 class KoumparasTester {
       public static void main(String[] a){
             Koumparas kou = new Koumparas();
             try {
                   kou.addKerma(Kerma.euro1);
                   kou.addKerma(Kerma.lepto20);
                   kou.addKerma(Kerma.lepto5);
                   kou.addKerma(Kerma.euro2);
                   System.out.println("Πλήθος        κερμάτων:        "        +
kou.getNumberOfCoins());
                   System.out.println("Αξία        κερμάτων:        "        +
kou.getTotalValue());

                   kou.doBreak();
                   kou.addKerma(Kerma.euro2);
                   System.out.println("Πλήθος        κερμάτων:        "        +
```

```
kou.getNumberOfCoins());
                    System.out.println("Αξία         κερμάτων:          "          +
kou.getTotalValue());
              } catch (Exception e) {
                    System.out.println(e);
              }

       }
 }
```

Βαθμολόγηση
Α) 10 μονάδες
4 μονάδες αν οι υπογραφές επαρκούν
2 μονάδες αν έχουν γράψει και pre/post, inv
2 αν έχουν καθορίσει και exception

Β) 10 μονάδες:
6 μονάδες: ορθότητα
4 μονάδες: συμφωνία με α)

Γ) 5 μονάδες