|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Opleiding**  **Titel Toetseenheid**  **Titel Toets**  **Code**  **Toetsperiode/studiejaar**  **Maximale score** | | | HBO-ICT  Programmeren Gevorderd  Programmeren Gevorderd Toets  TE.ICT.Vt.P.17.1.PRGV  P3 2017/2018  90 punten | | **Toetsvorm**  **Datum**  **Examinator**  **Nakijker**  **Tijdsduur**  **Cijfer** | | Tentamen  11 april 2018  Jos Foppele  Karel Pieterson  2 uur  (#punten+10)/100 | |
| **Toegestane hulpmiddelen** | | | Aantekeningen, Presentatie | | | | | |
|  | | | | | | | | |
| Naam |  | | | | | Cijfer | |  |
|  |  |  | | Klas |  |
| Studentnummer | |
|  | |  | |  |  |  | |  |
|  | | | | | | | | |
| **1. Piraten en Ninjas** | | | | | | | (15 punten) | |
| public class Pirate : Human of Ninja : Human  {  //class variabelen  private List<Human> crew; //pirate  private Human target; //ninja    //constructor  public Pirate/Ninja(string name, int age, crew of target) {  this.name = name;  this.age = age;  this.crew = crew; //pirate  this.target = target; //ninja  }  Overerving: Human: 3 pt.  private/public: 3 pt.  constructor parameters: 2 pt.  constructor code: 2 pt.  IsOld methode: 2 pt.  override: 2 pt  GetCrew/GetTarget: 1 pt.  //overige methoden (en extra ruimte)  public override bool IsOld() {  return age > 40; //pirate  return age > 50; //ninja  }  public List<Human> GetCrew() { //pirate  return crew;  }  public Human GetTarget() { // ninja  return target;  }      } | | | | | | | | |
| **2. Een boom vol mensen** | | | | | | | (10+20 punten) | |
| public void Add(Human h) {  if (h.IsOlderThan(item)) //of (!item.IsOlderThan(h))  if (old == null)  old = new Tree(h);  else  old.Add(h);  else  if (young == null)  young = new Tree(h);  else  young.Add(h);  Correcte if (old/young): 4 pt.  Controle oud null: 1 pt.  Oud nieuwe tree bij null: 1 pt.  Oud.Add bij niet null: 1 pt.  Controle jong null: 1 pt.  Jong nieuwe tree bij null: 1 pt.  Jong.Add bij niet null: 1 pt.  } | | | | | | | | |
| public List<Human> DeathList() {  List<Human> result = new List<Human>();  if (item is Ninja)  result.Add(((Ninja)item).GetTarget());  if (young != null)  result.AddRange(young.DeathList());  if (old != null)  result.AddRange(old.DeathList());  Result list aanmaken en returnen: 5 pt.  Item toevoegen als Ninja: 3 pt.  Cast van item: 2 pt.  Correct gebruik is in if: 2 pt.  young controleren of null: 2 pt.  young toevoegen aan resultlist: 2 pt.  old controleren of null: 2 pt.  old toevoegen aan resultlist: 2 pt.  return result;  } | | | | | | | | |
| **3. Piratenschip vol plunder** | | | | | | | (15+15 punten) | |
| public class Ship {  private Pirate captain;  private Plunder loot = null;  public Ship(Pirate captain) {  this.captain = captain;  }  **Add: (6pt)**  Controle loot != null: 2 pt.  loot = new plunder bij null: 2 pt.  loot.Add bij niet null: 2 pt.  public void Add(string d, int v) {  if (loot == null)  loot = new Plunder(d, v);  else  loot.Add(d, v);  }  **CalculateValue: (9pt)**  Controle loot != null: 3 pt.  Return 0 bij geen loot: 3 pt.  Return loot.CalculateValue(): 3 pt.  public int CalculateValue() {  if (loot != null)  return loot.CalculateValue();  return 0;  }  } | | | | | | | | |
| public class Plunder {  private string description;  private int value;  private Plunder next = null;  public Plunder(string d, int v) {  this.description = d;  this.value = v;  }  **Add: (6pt)**  Controle next != null: 2 pt.  next = new plunder bij null: 2 pt.  next.Add bij niet null: 2 pt.  public void Add(string d, int v) {  if (next == null)  next = new Plunder(d, v);  else  next.Add(d, v);  }  public int CalculateValue() {  **CalculateValue: (9pt)**  Controle next != null: 3 pt.  Return value bij geen next: 3 pt.  Return value + next.CalculateValue(): 3 pt.  if (next == null)  return value;  return value + next.CalculateValue();  }  } | | | | | | | | |
| **4. Gouden ratio berekenen** | | | | | | | (15 punten) | |
| public double GoldenRatio(int n) {  if (n == 1) // 0 mag ook  return 1; // 1.0 mag ook, 0 mag niet!  else  return 1 + (1 / GoldenRatio(n-1));  Stopconditie, if n == 1: 3 pt.  Stopconditie, return 1: 3 pt.  Recursieve berekening: 7 pt.  Return 1 + recursieve berekening: 2 pt.  } | | | | | | | | |
|  | | | | | | | | |
| **Extra ruimte** | | | | | | | | |
|  | | | | | | | | |