Self-Documenting Code

Miloš Djurović 1160/2012

Dokumentacija

- **Spoljna dokumentacija**
- **Stil programiranja kao dokumentacija**
- **Komentarisati ili ne?**
- **Vrste komentara**
- **Zaključak**

Spoljna dokumentacija

- Dokumentacija na softverskom projektu sastoji se od informacija kako unutar koda tako i izvan istog - obično u obliku odvojenih dokumenata ili razvojnih projekata.
- "Na velikim projektima većina dokumentacija je van izvornog koda (Jones 1998)"
- Globalna definicija problema, zahteva i arhitekture

Stil programiranja kao dokumentacija

- Za razliku od spoljne dokumentacije interna dokumentacija se nalazi unutar samog programa.
- Mnogo detaljnija dokumentacija i na mnogo višem nivou apstrakcije
- Bliza konkretnim problemima i sugestijama
- Značajna u slučaju nadgradnje ili izmene koda.

Dobra struktura koda

- Glavna karakteristika dobrog koda
- Pregledan stil pisanja samog koda
- Jasno definisanje linije
- Dobri nazivi promenjivih, klasa, korišćenje imena i naziva umesto slova i brojeva.
- Smanjena mogucnost mešanja redova koda prilikom traženja konkretne stvari

Primer loše organizovanog koda

Java Example of Poor Documentation Resulting from Bad Programming Style

```
for ( i = 1; i <= num; i++ ) {
meetsCriteria[ i ] = True;
}
```

```
for ( i = 2; i <= num / 2; i++ ) {
j = i + i;
while ( j <= num ) {
meetsCriteria[ j ] = False;
j = j + i;
}
}
for ( i = 1; i <= num; i++ ) {
if ( meetsCriteria[ i ] ) {
System.out.println ( i + " meets criteria." );
}
}
```

Primer dobro organizovanog koda

Java Example of Documentation Without Comments (with Good Style)

```
for ( primeCandidate = 1; primeCandidate <= num; primeCandidate++ ) {
   isPrime[ primeCandidate ] = True;
}

for ( int factor = 2; factor < ( num / 2 ); factor++ ) {
   int factorableNumber = factor + factor;
   while ( factorableNumber <= num ) {
      isPrime[ factorableNumber ] = False;
      factorableNumber = factorableNumber + factor;
   }
}

for ( primeCandidate = 1; primeCandidate <= num; primeCandidate++ ) {
   if ( isPrime[ primeCandidate ] ) {
      System.out.println( primeCandidate + " is prime." );
   }
}
```

Komentarisati ili ne?

- "Dobri komentari ne ponavljaju kod ili objašnjvaju. Oni služe da objasne viši nivo apstrakcije samog koda"
- "Oni su bacanje resursa(vremena i slova na tastaturi), dobar programer nema potrebe da se samo dokumentuje, sve što treba da ostavi kao rad nalazi se u samom kodu.“
- “Komentari u većini slučajeva samo izazivaju probleme, i ne treba ih pisati bez mnogo iskustva.Ako ih ne napišete kako treba doneće više štete nego pomoći"

Komentarisati ili ne?

- “Komentari nikada ne mogu da budu precizni kao sam jezik, zato su i nepotrebni"
- “Činjenica da postoje losi komentari ne znaci i da je komentarisanje lose"
- “Pisanje komentara te uči da dublje razmisljaš o problemu kroz činjenicu da i neko drugi treba da razume kod“
- “Trošenje vremena na komentare znači dodatno izbubljeno vreme koje je moglo da doprinese razvoju samog koda"

Vrste komentara

- Ponavljanje koda
- Objasnjenje koda
- Marker u kodu
- Summa summarum koda

Ponavljanje koda

- Ponavljanje koda kroz komentare razlicitim recima
- Samo dodaje onome ko čita vise posla umesto dodatnih informacija.
- Stvara utisak komplikovanosti koda

Objašnjenje koda

- Koristi se obicno za objasnjenje komplikovanih delova koda, trikova, ili osetljivih delova koda koji se mogu shvatiti na vise nacina.
- U nekim situacijama je veoma upotrebljivo ali negde samo može da uvede zabunu.
- Nekada je bolje kod napisati duže i detaljnije nego stavljati obimne komentare.
- Napisati kod što je moguće jasnije a komentari dodju samo kao kruna razumevanja eventualno nekih kraćih komplikovanih delova.

Marker u kodu

- Poruka sa leve strane koju čitamo pre samog koda.
- Služi da nam skrene pažnju da nešto u narednom delu koda ne radi, nije dovršeno ili nije provereno.
- Ove komentare bi trebalo posebno obeležavati, drugim fontom, slovima ili nekom specifičnom naznakom

Summa summarum koda

- Komentar koji daje sažetak koda u nekoliko linija odnosno jedna do dve rečenice.
- Korisni su jer pomažu lakšem snalaženju u velikim kodovima i oblastima.
- Pomažu bržem nalaženju problema u kodu.

Komentarisati efikasno i efektivno

- Efikasni kao i efektivni komentari nisu oduzimanje vremena.
- Komentar treba dalje omogućiti da bude promenjiv odnosno da moze lako i brzo da se ukloni ili ostavi novi.
- Komentar koji ne može lako da se prepravi obično biva ostavljen iako je kod promenjen.
- Ne treba gubiti mnogo vremena na ulepšavanje ili unošenje mnoštva ponavljajućih specijalnih znakova da bi skrenuli pažnju na komentar, dovoljno je ubaciti nekoliko uzvičnika ili zvezda na tastaturi.

Komentarisati efikasno i efektivno

- Ubacivanje previse tačaka ili čak crtanje interfejsa pomoću specijalnih karaktera samo govori o tome da se mnogo vremena gubi i smanjuje efikasnost.
- Početkom 1980-ih na jednom IBM test PC računaru primećeno je drastično sporije izvršavanje programa zbog mnoštvo komentara u kodu. Tada je to bilo mnogo uticajnije zbog manjka memorije računara, naročito prilikom izvršavanja programa preko mreže gde je mreža usko grlo sistema.

Optimalan broj komentara

- Capers Jones. 2000. godina
- Na svakih 10 linija koda u proseku dodje jedan komentar.
- Redje komentarisanje bi mozda dovelo do slabijeg razumevanja koda.
- A više do prenatrpanosti.

Zaključak

- Loše komentarisanje je gubljenje vremena a veoma često i pogrešno razumevanje koda.
- Dobar kod je najbolji komentar, naravno kako ne postoji apsolutno savršen kod, uvodimo komentare.
- Pre nego što uvedeš komentar pogledaj još jednom kod, možda ga je moguće unaprediti.
- Komentari služe da nam kažu one stvari koje kod samim čitanjem ne moze, ili bar ne može u razumnom vrmenskom roku.